NTNU

Norwegian University of
Science and Technology

Optimal Operation of Heat Exchanger Networks

Daniel Greiner Edvardsen

Chemical Engineering and Biotechnology
Submission date: June 2011
Supervisor: Sigurd Skogestad, IKP

Norwegian University of Science and Technology
Department of Chemical Engineering

Abstract

Optimal operation of heat exchanger networks is much less studied than optimal design. The control objective of a heat exchanger network is to control the temperature out of the heat exchanger network within a certain range. Often the control objective is to maximize the end temperature, and this is the focus of this study. A typical example is feed preheating in a crude oil fractionator. There seem to be no simple systematic ways to maximize the end temperature and the practical solutions are often suboptimal. Alternatively, RTO is used which is both challenging and expensive.

This study investigates the performance of a self-optimizing control strategy proposed by PhD candidate Johannes Jäschke. The method maxmimizes the end temperature and relies only on cheap temperature measurements, i.e. no flow measurements or technical data (heat exchanger area, heat transfer coefficients, heat capacities etc.) are necessary. The method has been demonstrated in four different cases; two theoretical cases and two real cases. The real cases are from Perstorp in Perstorp, Sweden and Statoil Mongstad outside of Bergen, Norway. A dynamic model has been made for the Perstorp case. Also, it has been looked into if the same self-optimizing control strategy can be used for an LNG process.

The self-optimizing control strategy perform well in all the cases investigated. At Perstorp it improves the performance, while at Statoil Mongstad the performance is just as good as the existing RTO. The results are presented for Perstorp and Statoil Mongstad which both are optimistic about the method.

Key words: optimal operation, heat exchanger networks, feedback control, crude oil fractionator, self-optimizing control, null space method, LNG, C3MR

1 Preface

This master thesis was written during the spring of 2011 as a compulsory part of the master degree within Chemical Engineering at NTNU.

After five and a half year at the university it was with great motivation I started on my final work as a student. After having a short conversation with my co-supervisor Johannes Jäschke I ended up with the project *Optimal operation of heat exchanger networks*. To be honest, I did not find the topic very exciting to begin with, but that has definitely changed throughout the semester.

I have learned alot from working with Johannes; about heat exchange, MATLAB, LaTeX and self-optimizing control. Thank you very much for being so helpful.

I feel honoured that I have had the opportunity to work with Sigurd Skogestad this year as a master student. With your level of knowledge, in addition to being so kind and helpful, it is always a pleasure to come by your office. I hope we get the opportunity to cooperate in the future as well. Thank you very much!

I would also like to thank all the friends I have got to know during these six years as a student. You have made them the best years of my life! Espescially Jonas Solbakken at University of Bergen, Tanner D. Johnston at University of Alberta and Tom-Gøran Skog at NTNU. Thank you!

Daniel

I declare that this is an independent work according to the exam regulations of the Norwegian University of Science and Technology

Date and signature:

.....

Contents

List of Figures

List of Tables

Symbol	Explanation	Unit
ΔT_{AM}	Arithmetic mean temperature difference	[K]
ΔT_{LM}	Logarithmic mean temperature difference	[K]
ΔT_{sup}	Amount of superheating	[K]
ε	Effectiveness	[–]
λ	Langrangian multiplier vectors	[–]
ρ	Fluid density	[g/cm^3]
ω_i	Heat capacaty rate of stream i	[W/K]
Θ	Temperature difference at heat exchanger ends	[K]
A	Area	[m^2]
c_p	Heat capacity	[J/kgK]
c	Controlled variable	[°C]
c_s	Setpoint for controlled variable	[°C]
$\mathbb{C}^*$	Set of complex numbers without zero	[–]
d	Disturbance	[various]
F	Sensitivity matrix	[–]
g	Equality constraint vector	[various]
h	Inequality constraint vector	[various]
h	Heat transfer coefficient	[W/m^{2}K]
H	Measurement combination matrix	[–]
J	Cost function	[°C]
$J_{opt}(d)$	Reoptimizend cost function at full disturbance	[°C]
$J(u,d)$	Cost function at full disturbance d	[°C]
$J_{z,red}$	Reduced gradient	[–]
k	Thermal conductivity	[W/mK]
L	Wall thickness	[m]
$L(u,d)$	Loss at full disturbance d	[°C]
MV	Manipulated variables	[–]
m	Mass flow rate	[kg/s]
N	Model order	[–]
N	Number of shells	[–]
NTU	Number of transfer units	[–]
Q	Heat transfer	[W]
$\mathcal{R}$	Sparse resultant	[–]

Continued

Symbol	Explanation	Unit
T	Temperature	[°C]
$T_{h,i}^{in}$	Hot temperature in of stream i	[°C]
$T_{h,i}$	Hot temperature out of stream i	[°C]
T_i	Cold temperature out of stream i	[°C]
T_0	Cold inlet temperature	[°C]
u	Split (mass fraction)	[–]
U	Overall heat transfer coefficient	[W/m^{2}K]
V	Volume	[m^3]
y	Measured variable	[various]

2 Introduction

Optimization of heat exchanger networks (HENs) has been studied extensively the last 30 years (Gorji-Bandpy et al., 2011) and is a typical problem in process design. There are three conventional methods available to help constructing the network, including Pinch Analysis Method (Linnhoff and Hindmarsh, 1983), Mathematical Programming Method (Bjork and Nordman, 2005) and Meta-heuristic Optimization Methods (Lin and Miller, 2004). All of these methods aim to design a reasonable trade-off between capital cost and operating cost (Glemmestad et al., 1999). When Linnhoff and Hindmarsh (1983) proposed the Pinch Design Method, an approach which sets up guidelines to design HENs, they established a theoretical foundation for process integration which is found in many chemical engineering text books (Novazzi and Zemp, 2009). However, Bjork and Nordman (2005) wrote that the advantage with mathematical programming methods is that a rigorous optimization of the structure, heat exchanger sizes and utility usage can be carried out, whilst in the Pinch Analysis Method the designer must make these decisions. The meta-heuristic optimization approach described in Lin and Miller (2004) uses a stochastic optimization approach which uses adaptive memory.

According to Glemmestad et al. (1999), the total design effort required for a HEN typically involves three steps:

1. *Nominal design*: Synthesize one or more networks with good properties for nominal stream data.

2. *Flexibility and controllability*: Investigate the networks with regard to flexibility and controllability.

3. *Operation*: Design a control system to operate the HEN properly. Involves control structure selection and possibly some method for on-line optimization.

This study focuses on the last step, operation of HENs, which is much less studied compared to finding nominal and flexible HENs (Glemmestad et al., 1999). Bypass selection for control of HENs was investigated by Mathisen et al. (1992), without considering utility consumption. A method for operation of HENs that minimizes utility consumption is proposed by Mathisen et al. (1994b). Repeated steady state

optimization is presented by Boyaci et al. (1996) and a method for on-line optimization and control of HENs by Aguilera and Marchetti (1998).

The control objective may be to control the temperature out of the heat exchanger network within a certain range, but often it is to maximize the end temperature. Two typical examples are air preheating in a power plant and feed preheating in a crude oil fractionator. Stream splits are typical in HENs, see Figure 2.1.

To save energy, the objective could be to maximize the end temperature T_{end} or, in other words, maximize the heat transfer $Q_1 + Q_2$, where Q indicates the heat transfered from hot to cold side for HX1 and HX2, respectively. This is the problem that will be studied in this report.

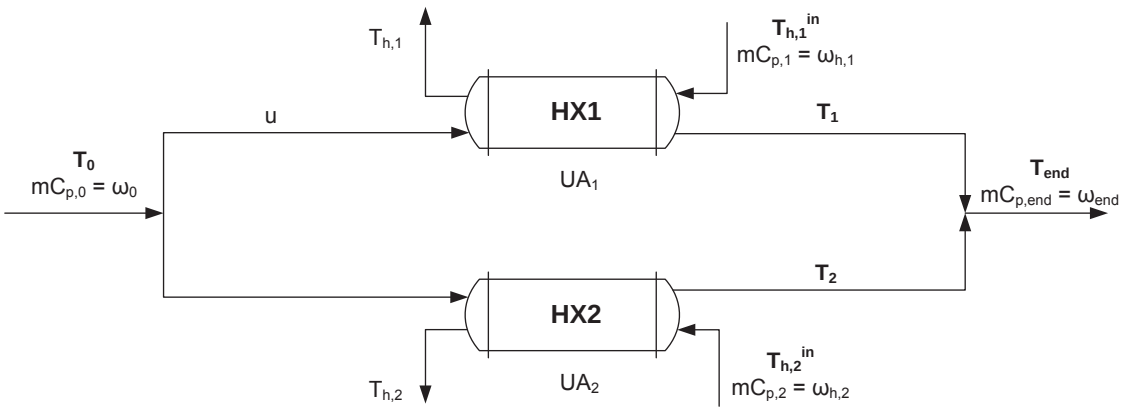


Figure 2.1: Heat exchanger network involving stream split

According to Jäschke and Skogestad (2011b) there seem to be no simple systematic ways to maximize T_{end}. Usually one of the folling is done in practice:

- Keep the split u constant
- Keep either T_1 or T_2 constant
- Let the operators adjust the split according to some heuristics
- Use real-time optimization

The first two methods are suboptimal, while the third can be optimal, but requires some effort from the operators. Using RTO is challenging and expensive because of difficulties in building and adapting accurate models for complex chemical processes

(Chachuat et al., 2009) and since it involves steady state detection, state/parameter estimation, data reconcilation and solving of a nonlinear optimization problem online (White, 1997).

By using resultants to eliminate unknown variables, Jäschke and Skogestad (2011b) have come up with a unique control strategy to maximize (or minimize) the end temperature out of HENs involving splits. The method is further described in Chapter 4. It does not require any model of the HEN, only simple temperature measurements. Four different cases have been simulated to investigate the performance of the method and these are described in Chapter 5, Chapter 6 and Chapter 7.

3 Heat Exchange and Self-Optimizing Control

In this chapter the reader will be given a brief introduction to the relevant topics for this study. First, the concept of heat exchange will be explained and then self-optimizing control will be introduced.

3.1 Heat Exchange

3.1.1 Modelling of Heat Exchangers

Heat exchange is a process where the purpose is to transfer heat Q to or from a process stream (Skogestad, 2003a). This heat transfer is conducted in a *heat exchanger* where the heat is transfered from hot to cold side through the heat exchanger's wall. The hot and the cold stream can flow co-current or countercurrent, where countercurrent (see Figure 3.1) is the most effective. In shell-and-tube heat exchangers both co-current and countercurrent flow exist. Other types of heat exchangers are spiral plate heat exchangers, kettle type heat exchangers, plate fin heat exchangers and sprial wound heat exchangers.

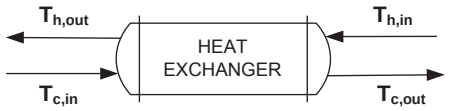


Figure 3.1: Ideal countercurrent heat exchanger

Assuming a countercurrent heat exchanger and constant hot and cold inlet temperatures, the heat transfer Q can be calculated according to the heat exchanger model

$$Q = UA\Delta T_{LM} \tag{3.1}$$

where U is the overall heat transfer coefficient [W/K] and A is the heat exchanger area [m^2]. The overall heat transfer coeffiecient is the reciprocal of the overall

resistance to heat transfer. If the fouling factors and tube wall thermal resistance are neglected, and the hot and cold heat exchanger areas are assumed to be equal, U is given by (Incropera et al., 2007):

$$U = \frac{h_c h_h}{h_c + h_h} \tag{3.2}$$

where h_h and h_c are the heat transfer coefficient in $\mathrm{W/m^2K}$ for hot and cold side, respectively.

ΔT_{LM} is the logarithmic mean temperature difference (LMTD) for countercurrent flow and is given by

$$\Delta T_{LM} = \frac{(T_{h,in} - T_{c,out}) - (T_{h,out} - T_{c,in})}{\ln\left(\frac{T_{h,in} - T_{c,out}}{T_{h,out} - T_{c,in}}\right)} = \frac{\Theta_1 - \Theta_2}{\ln\left(\frac{\Theta_1}{\Theta_2}\right)} \tag{3.3}$$

where the temperature differences Θ_1 and Θ_2 between hot and cold side for ideal countercurrent flow have been introduced. The energy balance for the heat exchanger in Figure 3.1 is

$$Q = m_c c_{p,c}(T_{c,out} - T_{c,in}) \tag{3.4}$$

$$Q = m_h c_{p,h}(T_{h,in} - T_{h,out}) \tag{3.5}$$

where constant heat capacities c_p [J/kgK] are assumed, i.e. $h = c_p T$. Summarized, the steady state balance for heat exchangers can be expressed by three equations:

$$Q = UA\Delta T_{LM} = m_c c_{p,c}(T_{c,out} - T_{c,in}) = m_h c_{p,h}(T_{h,in} - T_{h,out}) \tag{3.6}$$

In most of the case studies countercurrent flow will be assumed. However, in the industry shell-and-tube exchangers are by far the most common type of heat transfer equipment. Shell-and-tube heat exchangers are used in one of the case studies in this work.

3.1.2 Shell-and-tube Heat Exchangers

Some reasons for using shell-and-tube heat exchangers are (Sinnott and Towler, 2009):

1. The configuration gives a large surface area in a small volume
2. Good mechanical layout
3. Well-established fabrication techniques
4. Can be constructed from a wide range of materials
5. Easily cleaned
6. Well established design procedures

The heat exchanger consists of a bundle of tubes in a cylindrical shell. The flow pattern is then a combination of countercurrent and cocurrent flow and the heat exchanger model is given by

$$Q = UA\Delta T_{LM}F \tag{3.7}$$

where the LMTD correction factor F is introduced. $F = 1$ for true countercurrent flow, and is less than 1 when shell-and-tube heat exchangers are used. Configurations where $F < 0.8$ should not be used because the heat transfer is relatively inefficient. The correction factor can be calculated analytically as follows (Serth, 2007):

$$R = \frac{T_a - T_b}{t_b - t_a} \tag{3.8}$$

$$P = \frac{t_b - t_a}{T_a - t_a} \tag{3.9}$$

$$\alpha = \left(\frac{1 - RP}{1 - P}\right)^{1/N} \tag{3.10}$$

$$S = \frac{\alpha - 1}{\alpha - R} \tag{3.11}$$

$$F = \frac{\sqrt{R^2 + 1}\ \ln\left(\frac{1-S}{1-RS}\right)}{(R-1)\ln\left[\frac{2-S(R+1-\sqrt{R^2+1})}{2-S(R+1+\sqrt{R^2+1})}\right]} \tag{3.12}$$

where
T_a = inlet temperature of shell-side fluid
T_b = outlet temperature of shell-side fluid
t_a = inlet temperature of tube-side fluid
t_b = outlet temperature of tube-side fluid
N = Number of shells

If $R = 1$ the correction factor is given by

$$S = \frac{P}{N - (N-1)P} \tag{3.13}$$

$$F = \frac{S\sqrt{(2)}}{(1-S)\ln\left[\frac{2-S(2-\sqrt{2})}{2-S(2+\sqrt{2})}\right]} \tag{3.14}$$

It should be noted that F is independent of which fluid is in the shell and which is in the tubes, but R and P are not.

In this work the measumrent relations are not used. Hence, we can only eliminate n_g number of variables.

3.1.3 Approximation

In Equation (3.1) the LMTD was introduced for countercurrent heat exchangers. When $\Theta_1 = \Theta_2$ the LMTD becomes singular and is defined as being equal to the arithmetic mean temperature difference (AMTD) ΔT_{AM}. Since ΔT_{AM} is defined for any combination of Θ_1 and Θ_2 and does not include the logarithmic term it is often better to use than ΔT_{LM} for numerical calculations. Skogestad (2003a) notes that if $1/1.4 < \Theta_1/\Theta_2 < 1.4$ the error is less than 1% if LMTD is replaced by the AMTD:

$$\Delta T_{LM} = \frac{\Theta_1 - \Theta_2}{\ln\left(\frac{\Theta_1}{\Theta_2}\right)} \approx \frac{\Theta_1 + \Theta_2}{2} = \Delta T_{AM} \tag{3.15}$$

The approximation error for different selections of Θ_1 and Θ_2 is presented in Figure 3.1.3.

The figure shows that when Θ_1 and Θ_2 are similar the error is close to zero. However, if Θ_1/Θ_2 is much smaller or larger than 1 the error can become significant. A

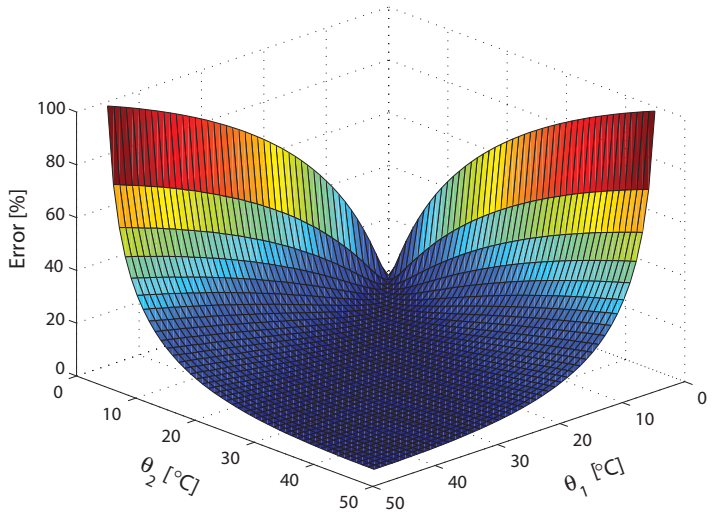


Figure 3.2: Error by approximating ΔT_{LM} with ΔT_{AM}

literature search reveals that other approximations for LMTD have been found that are associated with smaller approximation errors than the AMTD. Three approxmations by Chen (1987), Underwood (1933) and Paterson (1984) are, respectively:

$$\Delta T_{\mathrm{CM1}} = \sqrt[3]{\left(\frac{\Theta_1\Theta_2^2 + \Theta_1^2\Theta_2}{2}\right)} \tag{3.16}$$

$$\Delta T_{\mathrm{UM}} = \left(\frac{\Theta_1^{1/3} + \Theta_2^{1/3}}{2}\right)^3 \tag{3.17}$$

$$\Delta T_{\mathrm{PM}} = \frac{2}{3}\Delta T_{GM} + \frac{1}{3}\Theta_{AM} \tag{3.18}$$

where $\Delta T_{GM} = \sqrt{\Theta_1\Theta_2}$ is the geometric mean temperature difference. Chen also comes up with a "tuned" version of Underwood's approximation:

$$\Delta T_{\mathrm{CM2}} = \left(\frac{\Theta_1^{0.3275} + \Theta_2^{0.3275}}{2}\right)^{1/0.3275} \tag{3.19}$$

Plotting the error of all these approximations in contour plots for different selections of Θ_1 and Θ_2 reveals which approximation is the better. The plot is presented in Figure 3.3 where the error is defined as

$$\text{Error} = \frac{\Delta T - \Delta T_{LM}}{\Delta T_{LM}} \tag{3.20}$$

where ΔT represents the different approximations. The size of the error is indicated with different colours.

From the figure it is obvious that Chen's tuned version of Underwood's approximation is the best (and most complicated) approximation and the AMTD is the worst (and simplest) approximation.

In Chapter 4 the self-optimizing control strategy will be presented. The approach makes use of sparse resultants which only works with polynomials. Hence, Paterson's approximation can not be used, but the rest are candidate approximations.

3.1.4 Steady State Simulation of Heat Exchangers

When the heat exchanger area is given and we want to simulate the heat exchanger, the ε-NTU method can be applied. The equations used are the same as the energy balances in Equation (3.6) (for countercurrent heat exchangers) and Equation (3.8)-(3.12) (for shell-and-tube heat exchangers), just written in another way. The method is easy to use and good for simulations. The *number of transfer units* is defined as

$$NTU \equiv \frac{UA}{C_{min}} \tag{3.21}$$

where $C_{min} = \min\{m_c c_{p,c}, m_h c_{p,h}\}$. If countercurrent flow is assumed the effectiveness ε is given by (Incropera et al., 2007):

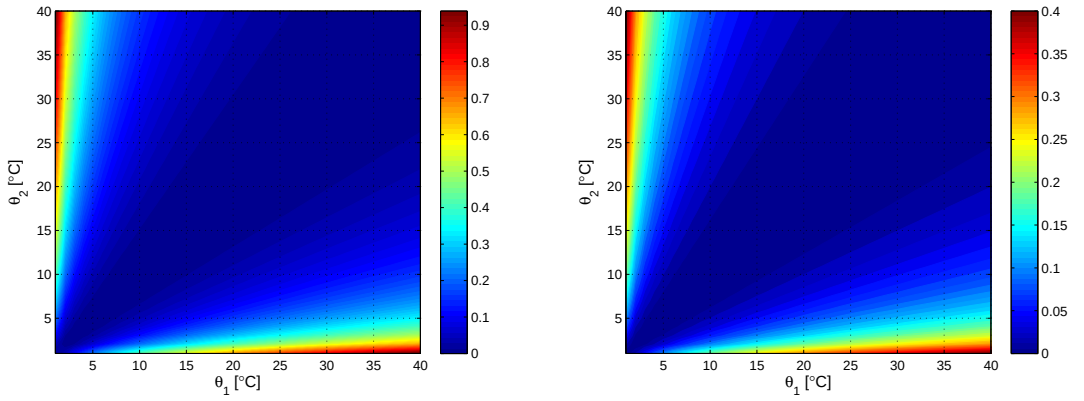
$$\varepsilon = \frac{1 - \exp(-NTU(1 - C_r))}{1 - C_r \exp(-NTU(1 - C_r))} \tag{3.22}$$

where $C_r \equiv C_{min}/C_{max}$ and $C_{max} = \max\{m_c c_{p,c}, m_h c_{p,h}\}$. If $C_r = 1$ Equation (3.22) becomes singular and cannot be used. ε is then given by

$$\varepsilon = \frac{NTU}{1 + NTU} \tag{3.23}$$

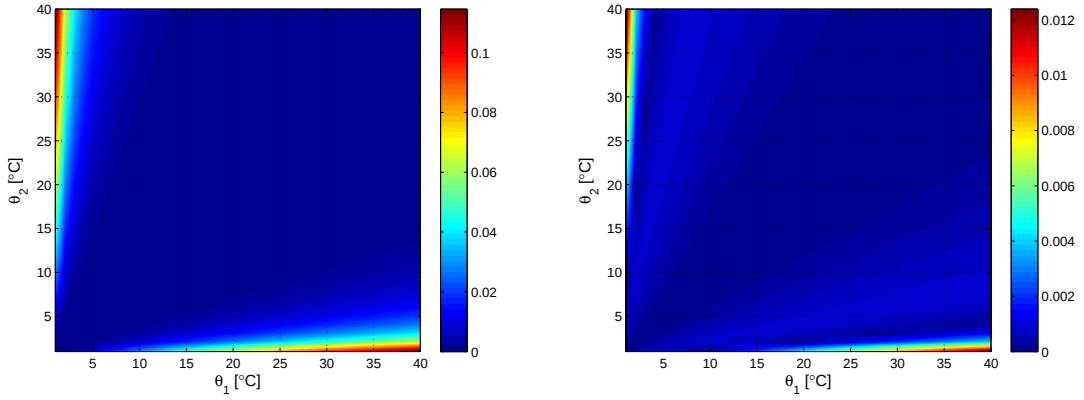
for countercurrent flow.

When shell-and-tube heat exchangers are used the effectiveness can be found by



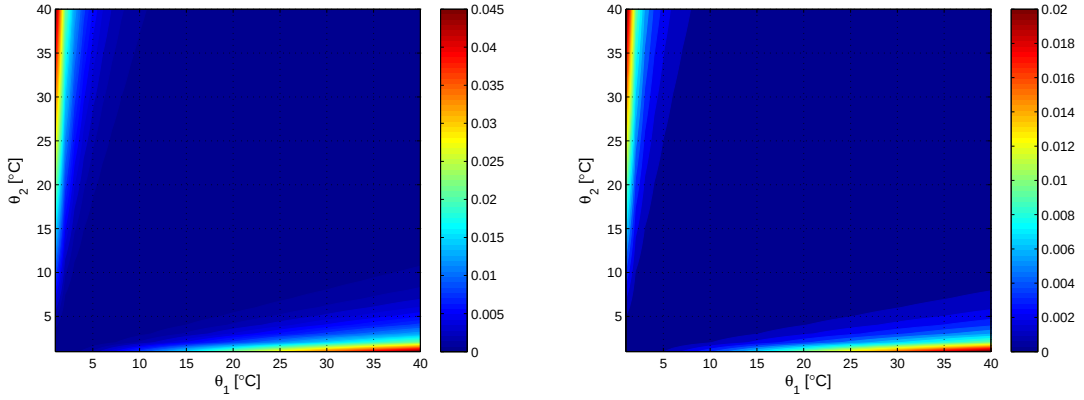
(a) Arithmetic

(b) Geometric



(c) Chen

(d) Chen Tuned



(e) Paterson

(f) Underwood

Figure 3.3: Error associated with approximating ΔT_{LM}

$$\varepsilon = \frac{\left(\frac{1-\varepsilon_1 C_r}{1-\varepsilon_1}\right)^n - 1}{\left(\frac{1-\varepsilon_1 C_r}{1-\varepsilon_1}\right)^n - C_r} \tag{3.24}$$

where n is number of shells and ε_1 is the effectiveness for one shell pass:

$$\varepsilon_1 = \frac{2}{1 + C_r + \sqrt{1+C_r^2} \times \frac{1+\exp\left[-(NTU)_1\sqrt{1+C_r^2}\right]}{1-\exp\left[-(NTU)_1\sqrt{1+C_r^2}\right]}} \tag{3.25}$$

In Equation (3.25) it is assumed that the total NTU is equally distributed between shell pasees of the same arrangement, i.e. $(NTU)_1 = NTU/n$.

The hot and cold temperature out can then be found:

$$T_{h,out} = \left(1 - \frac{C_{min}\varepsilon}{m_h c_{p,h}}\right) T_{h,in} + \frac{C_{min}\varepsilon}{m_h c_{p,h}} T_{c,in} \tag{3.26}$$

$$T_{c,out} = \frac{C_{min}\varepsilon}{m_c c_{p,c}} T_{h,in} + \left(1 - \frac{C_{min}\varepsilon}{m_c c_{p,c}}\right) T_{c,in} \tag{3.27}$$

Thus, if the product UA, hot and cold heat capacities, inlet temperatures and mass flows are known, the outlet temperatures can be calculated for both countercurrent and shell-and-tube heat exchangers. According to Equation (3.26) and Equation (3.27), the ε-NTU method yields a linear relationship between the inlet temperatures and the outlet temperatures. Note, however, that the outlet temperature is nonlinearly dependent on the flow rate.

3.1.5 Dynamic Modelling of Heat Exchangers

Dynamic models are central in the subject of process dynamics and control and are used to (Seborg et al., 2003):

- Improve understanding of the process
- Train plant operating personnel
- Develop a control strategy for a new process

- Optimize process operating conditions

In this study a dynamic model has been obtained in one of the case studies. An introduction to dynamic heat exchanger models is given below and the dynamic model is presented in Chapter 6.3.

A heat exchanger can be approximated by a lumped model where each fluid is modelled as mixed tanks in series (Mathisen et al., 1994a). The lumped model is presented in Figure 3.4.

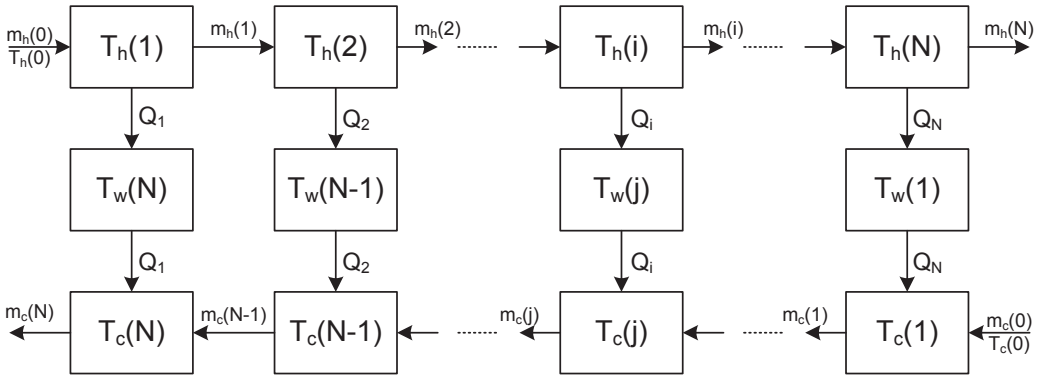


Figure 3.4: Lumped model of heat exchanger (Mathisen et al. (1994a), modified)

Here, negligible heat loss, constant heat capacity, constant densities, negligible pressure drop and equally distributed heat exchanger area A and volume V over the N cells are assumed. The differential equations resulting from the energy balance are:

$$\frac{dT_h(i)}{dt} = \left(T_h(i-1) - T_h(i) - \frac{h_h A}{\omega_h N}\Delta T_h(i)\right)\frac{m_h N}{\rho_h V_h} \tag{3.28}$$

$$\frac{dT_w(j)}{dt} = \left(h_h \Delta T_{w,h}(j) - h_c \Delta T_{w,c}(j)\right)\frac{A}{\rho_w c_{p,w} V_w} \tag{3.29}$$

$$\frac{dT_c(j)}{dt} = \left(T_c(j-1) - T_c(j) - \frac{h_c A}{\omega_c N}\Delta T_c(j)\right)\frac{m_c N}{\rho_c V_c} \tag{3.30}$$

where all subsript h, c and w denotes hot fluid, cold fluid and wall, respectively. Further, T is temperature, h is the heat transfer coefficient, A is the heat exhcanger area, V is volume, N is the model order, ρ is the density, c_p is the heat capacity and ω is the heat capacity rate. A complete derivation of the equations can be found in

Mathisen et al. (1994a). According to the authors a model order of $N > 6$ is typical and in this study a model order of 10 is used to be able to predict the apparent lag with good accuracy.

3.2 Self-Optimizing Control

Self-optimizing control is when acceptable (close to optimal) operation is achieved with constant setpoints for the controlled variables. The introduction to this concept presented here is taken from Skogestad (2004).

The general optimization problem is to minimize a certain objective function subject to some constraints:

$$\begin{aligned} &\text{minimize} && J(x, u_0, d) \\ &\text{subject to} && g(x, u_0, d) = 0, \quad h(x, u_0, d) \leq 0 \end{aligned} \tag{3.31}$$

where J is the objective function (or cost function), x represents the state variables, u_0 the available degrees of freedom and d the disturbances. The equality constraints g include the model equations which link the independent variables u and d with the states x. The inequality constraints h are necessary because the system should have, for instance, positive temperatures and mass flows. Some of the inequality constraints are often active constraints, which means that they should be equal to zero. These active constraints should be controlled with a corresponding number of degrees of freedom. The main problem is then to decide what to control with the remaining unconstrained degrees of freedom, u. If the states x are eliminated using the model equations g the remaining unconstrained problem is

$$\min_u \quad J(u, d) = J(u_{opt}(d), d) \stackrel{\text{def}}{=} J_{opt}(d) \tag{3.32}$$

where $u_{opt}(d)$ is to be found and $J_{opt}(d)$ is the optimal value of the cost function. The solution if problems like this is extensively studied. We want to find which controlled variables c (which is a selected subset of the measured variables y) to keep constant at the optimal values c_{opt}. c_{opt} should be insensitive to the disturbances d to obtain operation close to optimal. "Close to optimal" means that there is a loss assosciated

with keeping the controlled variable constant, and this loss can be expressed as

$$L(u,d) = J(u,d) - J_{\text{opt}}(d) \tag{3.33}$$

The controlled variables can be single measurements or a combination of measurements. The advantage is that we do not need to continuously solve the optimization problem (as in real-time optimization, RTO) which can be costly both to install and maintain. Self-optimizing control is (Skogestad, 2000):

> ... when we can achieve an acceptable loss with constant setpoint values for the controlled variables whithout the need to reoptimize when disturbances occur.

To select the controlled variables, the following guidelines presented by Skogestad (2000) can be used:

1. c_{opt} should be insensitive to disturbances (minimizes the effect of disturbances)
2. c should be easy to measure and control accurately (reduces the implementation error)
3. c should be sensitive to changes in the (steady state) degrees of freedom, i.e. $J = f(c)$ should be flat (minimizes the effect of the implementation error)
4. For cases with more than one unconstrained degrees of freedom, the selected controlled variables should be independent (minimizes the effect of the implementation error)

An ideal self-optimizing variable, proposed among others by Halvorsen and Skogestad (1997), is the gradient of the cost function, $c_{ideal} = J_{\text{u}} = \frac{\partial J}{\partial u}$, which should be zero to ensure optimal operation for all disturbances. However, measurement of the gradient is usually not available, and computing it requires knowing the value of unmeasured disturbances. To find which variables are the best to keep constant (approximations of the gradient), different approaches can be used:

1. Exact local method

2. Direct evaluation of loss for all disturbances ("brute force")

3. Maximum (scaled) gain method

4. Null space method

Next, the null space method will be described. The exact local method is described in Chapter 8.3.1. For details about the two other methods it is referred to Skogestad and Postlethwaite (2005).

3.3 The Null Space Method

Instead of choosing single measurements as controlled variables, combinations of measurements can also be considered. Choosing *linear* combinations we get

$$c = \mathbf{H}y \tag{3.34}$$

where $\mathbf{H}$ is the measurement combination matrix (to be found) and y are all the measurements available (Skogestad and Postlethwaite, 2005). Keeping c constant may not be optimal because of the presence of implemenation error and disturbances. If the implemenation error is neglected, keeping $c = c_{\mathrm{s}}$ is optimal if c_{opt} is independent of the disturbances (i.e. $c_{\mathrm{opt}} = 0 \cdot d$). The individual measurements are still dependent on d:

$$y_{\mathrm{opt}} = \frac{\mathrm{d}y_{\mathrm{opt}}}{\mathrm{d}d} d = \mathbf{F}d \tag{3.35}$$

where $\mathbf{F}$ is the sensitivity matrix. By making small changes in d, y_{opt} and $\mathbf{F}$ can be found. Inserting Equation (3.35) in Equation (3.34), with $c = c_{\mathrm{opt}}$, yields

$$\begin{aligned} c_{opt} &= \mathbf{H}y_{opt} \\ 0 \cdot d &= \mathbf{HF}d \\ \mathbf{HF} &= 0 \end{aligned} \tag{3.36}$$

for all d. This is equivalent to $\mathbf{H}$ lying in the left null space of $\mathbf{F}$. This is possible if $n_y \geq n_u + n_d$, where n_y is the number of measurements, n_u the number of manipulated variables and n_d the number of disturbances (Skogestad and Postlethwaite, 2005).

Recently, a new derivation was found by Jäschke (2011), starting from the gradient J_u. If $c = J_u$, we have from Equation (3.34)

$$J_u = \mathbf{H}y \tag{3.37}$$

After all the active contraints are controlled the gradient can be approximated to first order as

$$J_u = \begin{bmatrix} J_{uu} & J_{ud} \end{bmatrix} \begin{bmatrix} u \\ d \end{bmatrix} \tag{3.38}$$

and the measurements y are

$$y = \mathbf{G} \begin{bmatrix} u \\ d \end{bmatrix} \tag{3.39}$$

Solving for $[u, d]^T$ gives

$$\begin{bmatrix} u \\ d \end{bmatrix} = \mathbf{G}^{-1} y \tag{3.40}$$

and inserting into Equation (3.38) gives

$$J_u = \begin{bmatrix} J_{uu} & J_{ud} \end{bmatrix} \begin{bmatrix} \mathbf{G} \end{bmatrix}^{-1} y \tag{3.41}$$

If $\mathbf{H}$ is selected as $\mathbf{H} = \begin{bmatrix} J_{uu} & J_{ud} \end{bmatrix} \begin{bmatrix} \mathbf{G} \end{bmatrix}^{-1}$ we end up with Equation (3.37). Controlling $c = \mathbf{H}y = 0$ results in zero loss. This shows that the nullspace method can be considered as formulating the optimality conditions and eliminating u and d using the measurements.

3.4 Self-Optimizing Control for Polynomial Systems

Extending the nullspace method described in Chapter 3.3 to polynomial systems, we get

$$c = H(y) \tag{3.42}$$

where $H(y)$ is now nonlinear. According to Jäschke and Skogestad (2011a), self-optimizing control has only been considered using linear process models and quadratic approximation of the cost function. If the cost function has a strong curvature at optimum, local analysis (i.e. linear) may not be sufficient and the controlled variables not self-optimizing. We now try to achieve (close-to) optimal operation using the null space method extended for polynomial systems. Only the most important aspects are mentioned here, for all the details and proofs it is referred to Jäschke and Skogestad (2011a).

By abuse of notation, all the active constraints $h = 0$ in Equation (3.31) are included in the equality constraint vector g. The optimization problem can then be written as

$$\begin{aligned} \text{minimize} \quad & J(z,d) \\ \text{subject to} \quad & g(z,d) = 0 \end{aligned} \tag{3.43}$$

where the manipulated input variables u and the internal state variables x are combined in $z = [u, x]^T$.

Let z be a feasible point of the optimization problem and assume all gradient vectors $\nabla_z g_i(z,d)$ and $\nabla_z h_i(z,d)$ associated with $g = 0$ and $h = 0$ are linear independent. If z is locally optimal, then there exists Langrangian multiplier vectors λ such that the first order optimality conditions are satisfied:

$$\begin{aligned} \nabla_z J(z,d) + [\nabla_z g(z,d)]^T \lambda = 0 \\ g(z,d) = 0 \end{aligned} \tag{3.44}$$

The expressions in Equation (3.44) could potentially be used for control as self-optimzing variables, but they contain unknown variables in x, d and λ which must be eliminated.

First, the Langrangian multipliers λ are eliminated. If $N(z,d)$ is defined such that $[N(z,d)]^T \nabla_z g(z,d) = 0$ (i.e. N is a basis for the null space of ∇g), then premultiplying Equation (3.44) with $[N(z,d)]^T$ results in

$$\begin{aligned}[N(z,d)]^T \left(\nabla_z J(z,d) + [\nabla_z g(z,d)]^T \lambda\right) &= [N(z,d)]^T \nabla_z J(z,d) + \underline{0}\lambda \\ &= [N(z,d)]^T \nabla_z J(z,d) \\ &\stackrel{\text{def}}{=} J_{z,red}\end{aligned} \tag{3.45}$$

where the reduced gradient $J_{z,red}$ has been introduced. Controlling $J_{z,red} = 0$ and $g(z,d) = 0$ fully specifies the system at the optimum, but the reduced gradient still contains unknown variables in x and d which must be eliminated.

The sparse resultant can be used for this purpose. Briefly said, the sparse resultant $\mathcal{R}$ is replacing the constraint $J_{z,red} = 0$ with $\mathcal{R} = 0$, as stated by Theorem 2 in Jäschke and Skogestad (2011a):

Theorem 2 (Nonlinear measurement combinations as controlled variables). *Given $\hat{d} = [d,x]^T \in (\mathbb{R})^{n_{\hat{d}}}$, and $n_y + n_g = n_{\hat{d}}$ independent relations $g(y,\hat{d}) = m(y,\hat{d}) = 0$ such that the system*

$$\begin{aligned} g(y,\hat{d}) &= 0 \\ m(y,\hat{d}) &= 0 \end{aligned} \tag{3.46}$$

has finitely many solutions for $\hat{d} \in (\mathbb{C}^*)^{n_{\hat{d}}}$. Let $\mathcal{R}(J^{(i)}_{z,red}, g, m)$, $i = 1 \ldots n_c$ be the sparse resultants of the n_c polynomial systems composed of

$$J^{(i)}_{z,red}(y,\hat{d}) = 0, \qquad g(y,\hat{d}) = 0, \qquad m(y,\hat{d}) = 0 \quad i = 1 \ldots n_c, \tag{3.47}$$

then controlling the active constraints, $g(y,\hat{d}) = 0$ and $c_i = \mathcal{R}(J^{(i)}_{z,red}, g, m) = 0$ $i = 1 \ldots n_c$, yields optimal operation throughout the region.

where $\mathbb{C}^*$ denotes the set of complex numbers without zero and $m(y,\hat{d})$ are the measure, relations. $m(y,\hat{d})$ is only used for elimination purposes since it does not

affect the optimal solution (which, in contrast, $g(y, \hat{d})$ does). This theorem states that the number of measurements (n_y) plus the number of model equations (n_g) must equal the number of unknowns ($n_{\hat{d}}$) in order to be able to eliminate the unkowns. Then we have one more equation than unknowns because of $J_{z,red}$.

It is worth noticing how the state variable x is combined in a common variable in two different ways for different purposes. In Equation (3.43)-(3.45) x and u are combined in $z = [u, x]^T$. In Equation (3.46) and Equation (3.47) x and d are combined in $\hat{d} = [d, x]^T$. z is used for optimization and $\hat{d}$ is unknown and should be eliminated.

Example (One disturbance) In this example we consider a system of two polynomials in one unkown variable d and one measurement relation $m(y, d) = 0$. At optimum we must have

$$\begin{aligned} J_{z,red} = [N(z,d)]^T \nabla_z J(z,d) = a_0(y) + a_1(y)d + a_2(y)d^2 = 0 \\ m(y,d) = b_0(y) + b_1(y)d = 0 \end{aligned} \tag{3.48}$$

The sparse resultant coincides in the case of univarite polynomials (polynomials of only one variable) with the classical resultant, which is the determinant of the Sylvester matrix. The Sylvester matrix of the system in Equation (3.48) is

$$Syl = \begin{bmatrix} a_0 & a_1 & a_2 \\ b_0 & b_1 & 0 \\ 0 & b_0 & b_1 \end{bmatrix} \tag{3.49}$$

and the restultant is

$$\mathcal{R} = \det(Syl) = a_0 b_1^2 - a_1 b_0 b_1 + a_2 b_0^2 \tag{3.50}$$

For a solution d to exist, $\mathcal{R}$ must equal zero. Since $m(y, d) = 0 \ \forall \ d$, controlling $\mathcal{R} = 0$ is equivalent to controlling $J_{z,red} = 0$, and the operation will be optimal. The unkown variable in the system in Equation (3.48) could also be the state variable x.

To do elimination with multivariate polynomials the package `multires` (Buse and Mourrain, 1998) for `Maple` is used. Examples are presented as Case Studies in Chapter 4.1 and Chapter 4.2.

4 Self-Optimizing Control for Heat Exchanger Networks

In this chapter self-optimizing control will be related to the operation of heat exchanger networks. Since the end temperature is to be maximized the optimization problem is

$$\begin{aligned} \underset{u}{\text{minmize}} \quad & J = -T_{end} \\ \text{subject to} \quad & g = 0 \end{aligned} \tag{4.1}$$

where g is the steady state model of the heat exchanger network and u are the available degrees of freedom. It is assumed that the hot streams are disturbances and not degrees of freedom, hence the number of degrees of freedom is equal to the number of splits in the heat exchanger network.

Using the self-optimizing approach described in Chapter 3.2 and the extended null space method for polynomial systems a good controlled variable can be found for different heat exchanger networks. In this work HENs with a maximum of two heat exchangers in series and an unlimited number of parallel heat exchangers will be studied. In the following, a HEN with two heat exchangers in parallel and a HEN with two heat exchangers in series and one in parallel will be studied.

4.1 Two Heat Exchangers in Parallel

Two heat exchangers in parallel with a common feed temperature of T_0 is presented in Figure 4.1.

The split u is the ratio of the respective mass flow with the total inlet mass flow. To find the model equations g in Equation (8.1) the energy balances must be put up. Assuming ideal mixing between stream 1 and stream 2 the resulting energy balance is simply $u\omega_0 T_1 + (1-u)\omega_0 T_2 - \omega_0 T_{end}$. To find the other model equations Equation (3.6) must be used.

If the LMTD was used the problem could be solved numerically, for example by a real-time optimizer. However, the `multires` package mentioned in Chapter 3.4

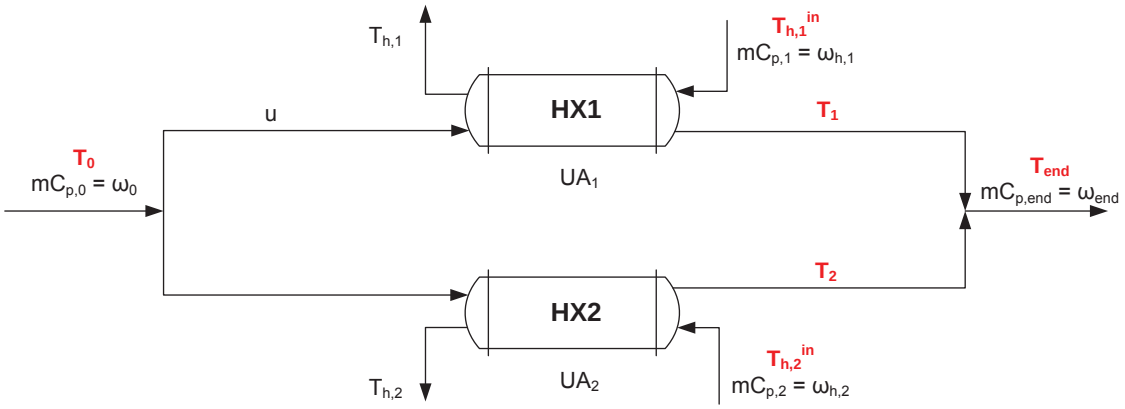


Figure 4.1: Two heat exchangers in parallel

requires polynomials as input. Six approximations was presented in Chapter 3.1.3, but only the AMTD will be used here. The main reasons are that the approximation is very simple and the error is small when Θ_1 and Θ_2 are similar. Also, it turns out that the self-optimizing variable achieved when this approximation is used is very simple as well. Thus, the heat exchanger model is

$$Q = UA\Delta T_{AM} \tag{4.2}$$

and the energy balance for the cold stream going through HX1 is:

$$U_1A_1\Delta T_{AM} = u\omega_0(T_1 - T_0) \tag{4.3}$$

$$0.5U_1A_1(T_{h,1}^{in} - T_1 + T_{h,1} - T_0) = u\omega_0(T_1 - T_0) \tag{4.4}$$

$$2u\omega_0(T_1 - T_0) - U_1A_1(T_{h,1}^{in} - T_1 + T_{h,1} - T_0) = 0 \tag{4.5}$$

where ω is the heat capacity rate mC_p. All the other symbols are explained in Figure 4.1. The three other energy balances are done in the exact same manner and not shown here. With all the energy balances in place, g can be written as

$$g = \begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ g_4 \\ g_5 \end{bmatrix} = \begin{bmatrix} u\omega_0T_1 + (1-u)\omega_0T_2 - \omega_0T_{end} \\ 2u\omega_0(T_1 - T_0) - U_1A_1(T_{h,1}^{in} - T_1 + T_{h,1} - T_0) \\ 2(1-u)\omega_0(T_2 - T_0) - U_2A_2(T_{h,2}^{in} - T_2 + T_{h,2} - T_0) \\ 2\omega_{h,1}(T_{h,1} - T_{h,1}^{in}) + U_1A_1(T_{h,1}^{in} - T_1 + T_{h,1} - T_0) \\ 2\omega_{h,2}(T_{h,2} - T_{h,2}^{in}) + U_2A_2(T_{h,2}^{in} - T_2 + T_{h,2} - T_0) \end{bmatrix} \tag{4.6}$$

The state variables $\mathbf{x}$ are the temperatures that varies:

$$\mathbf{x} = \begin{bmatrix} T_1 & T_2 & T_{end} & T_{h,1} & T_{h,2} \end{bmatrix} \tag{4.7}$$

and the manipulated variable is the split u. The combined vector $\mathbf{z}$ then becomes

$$\mathbf{z} = \begin{bmatrix} u & \mathbf{x} \end{bmatrix} = \begin{bmatrix} u & T_1 & T_2 & T_{end} & T_{h,1} & T_{h,2} \end{bmatrix} \tag{4.8}$$

Further, $\nabla_z J(z,d)$ is found to be

$$\nabla_z J(z,d) = \frac{\partial J}{\partial z} = \frac{\partial(-T_{end})}{\partial z} = \begin{bmatrix} 0 & 0 & -1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.9}$$

$\mathbf{N}(z,d)$ is calculated using `Maple`. For the complete `Maple` code see Appendix H.1. Now the reduced gradient $J_{z,red}$ from Equation (3.45) can be calculated (also using `Maple`):

$$\begin{aligned} J_{z,red} = [N(z,d)]^T \nabla_z J(z,d) = {} & U_2 A_2 U_1 A_1 T_1 \omega_{h,2} \omega_{h,1} - U_2 A_2 U_1 A_1 T_2 \omega_{h,2} \omega_{h,1} \ldots \\ & - U_2 A_2 U_1 A_1 T_2 \omega_{h,2} u \omega_0 + 2 U_1 A_1 \omega_0 T_0 u \omega_{h,2} \omega_{h,1} - 2 U_1 A_1 u \omega_0 T_1 \omega_{h,2} \omega_{h,1} \ldots \\ & - U_1 A_1 u \omega_0 T_1 U_2 A_2 \omega_{h,1} - U_1 A_1 \omega_0 T_0 U_2 A_2 \omega_{h,1} + \omega_0 T_0 U_1 A_1 u \omega_{h,2} U_2 A_2 \ldots \\ & - 2 U_2 A_2 T_2 \omega_{h,2} u \omega_0 \omega_{h,1} + U_1 A_1 \omega_0 T_0 u U_2 A_2 \omega_{h,1} + 2 \omega_0 T_0 u \omega_{h,2} U_2 A_2 \omega_{h,1} \ldots \\ & + U_1 A_1 \omega_0 T_1 U_2 A_2 \omega_{h,1} - 2 U_1 A_1 \omega_0 T_0 \omega_{h,2} \omega_{h,1} + 2 U_1 A_1 \omega_0 T_1 \omega_{h,2} \omega_{h,1} \end{aligned} \tag{4.10}$$

The temperatures marked with red in Figure 4.1 are assumed measured. The cold and hot inlet temperatures are normally measured along with the end temperature. In addition it is assumed that the cold outlet temperatures are measured. Thus, u, ω_0, $\omega_{h,1}$, $\omega_{h,2}$, U_1A_1 and U_2A_2 are unknowns and must be eliminated. The sparse resultant from Theorem 2 on page 18 is used for this, but since there are 6 unknowns and we only have 6 equations (5 model equations g and $J_{z,red}$) we can only eliminate 5 variables. Thus, only u, ω_0, $\omega_{h,1}$ and $\omega_{h,2}$ are eliminated using the `multires` package. The sparse resultant is found to be

$$\mathcal{R}(J_{z,red}, g) = -8 U A_1^2 U A_2^3 a_1 a_2 a_3 \tag{4.11}$$

where

$a_1 = (-T_{h,2}^{in} + T_2 - T_{h,2} + T_0)^2$

$a_2 = -T_{h,1}^{in} + T_1 - T_{h,1} + T_0$

$a_3 = T_0^2 T_{h,2}^{in} + 2T_0^2 T_1 - T_0^2 T_{h,1}^{in} - 2T_0^2 T_2 - 2T_0 T_{h,2}^{in} T_1 + T_0 T_2^2 - T_0 T_1^2 + 2T_0 T_2 T_{h,1}^{in} + T_{h,2}^{in} T_1^2 - T_2^2 T_{h,1}^{in}$

The system of equations has a solution if and only if $\mathcal{R} = 0$. The resultant is zero when any of the factors a_1, a_2 or a_3 are zero. Neither a_1 nor a_2 can be valid solutions since they (independently) contain no information about the other stream; $T_{h,1}^{in}$ is clearly important information when the split should be set, so keeping $a_1 = 0$ is not sufficient. The same argument holds for a_2. Thus, keeping a_3 equal to zero makes the resulatant zero and is a valid solution. Note that in Equation (4.11) neither $U_1 A_1$ nor $U_2 A_2$ appear in a_3 (luckily).

The resultant can further be simplified by moving the reference temperature. Until now there has been no restrictions on temperatue units (could be °C, K, F etc) which means that the reference temperature can be T_0 and then write all the other temperatures relative to T_0:

$$\Delta T_0 = 0$$

$$\Delta T_1 = T_1 - T_0$$

$$\Delta T_2 = T_2 - T_0$$

$$\Delta T_{h,1}^{in} = T_{h,1}^{in} - T_0$$

$$\Delta T_{h,2}^{in} = T_{h,2}^{in} - T_0$$

All terms in Equation (4.11) containing T_0 will then cancel and we end up with the self-optimizing controlled variable

$$c = \frac{\Delta T_1^2}{\Delta T_{h,1}^{in}} - \frac{\Delta T_2^2}{\Delta T_{h,2}^{in}} = 0 \tag{4.12}$$

Hence, for a heat exchanger network with two heat exchangers in parallel five measurements are needed to achieve self-optimizing control. More generally, two measurements are needed for each heat exchanger (cold temperature out and hot temperature in) plus the cold inlet temperature: $n_m = 2n_{HX} + 1$.

By abuse of notation the "Δ" is ommited in the report. The final self-optimizing controlled variable is then

$$c = \frac{T_1^2}{T_{h,1}^{in}} - \frac{T_2^2}{T_{h,2}^{in}} = 0 \tag{4.13}$$

4.2 Two Heat Exchangers in Series and One in Parallel

Two heat exchangers in series with one in parallel with feed temperature T_0 is presented in Figure 4.2.

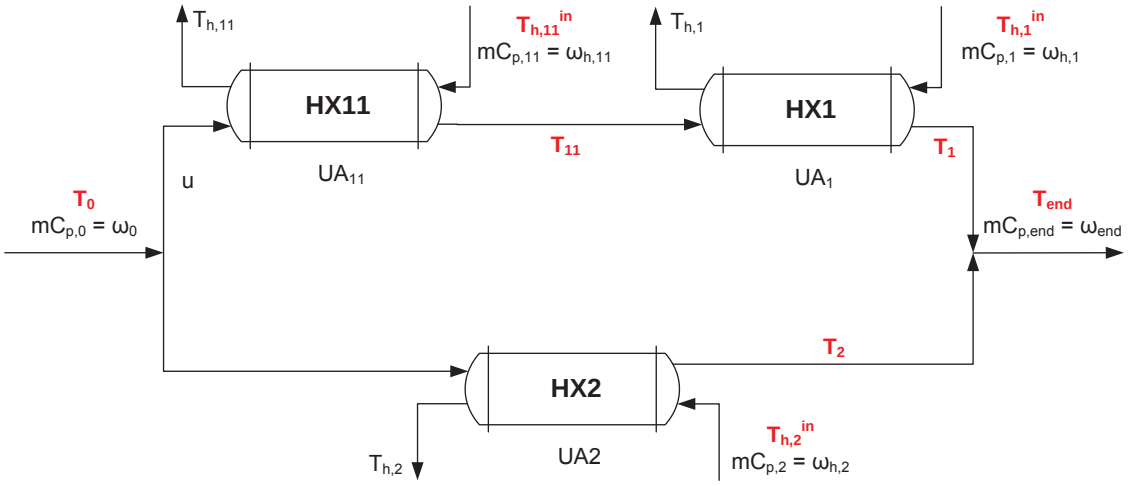


Figure 4.2: Two heat exchangers in series and one in parallel

As before, performing energy balances g in Equation (8.1) can be found (the derivations are not shown here):

$$g = \begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ g_4 \\ g_5 \\ g_6 \\ g_7 \end{bmatrix} = \begin{bmatrix} u\omega_0 T_1 + (1-u)\omega_0 T_2 - \omega_0 T_{end} \\ 2u\omega_0(T_{11} - T_0) - U_{11}A_{11}(T^{in}_{h,11} - T_{11} + T_{h,11} - T_0) \\ 2u\omega_0(T_1 - T_{11}) - U_1 A_1(T^{in}_{h,1} - T_1 + T_{h,1} - T_{11}) \\ 2(1-u)\omega_0(T_2 - T_0) - U_2 A_2(T^{in}_{h,2} - T_2 + T_{h,2} - T_0) \\ 2\omega_{h,11}(T_{h,11} - T^{in}_{h,11}) + U_{11}A_{11}(T^{in}_{h,11} - T_{11} + T_{h,11} - T_0) \\ 2\omega_{h,1}(T_{h,1} - T^{in}_{h,1}) + U_1 A_1(T^{in}_{h,1} - T_1 + T_{h,1} - T_{11}) \\ 2\omega_{h,2}(T_{h,2} - T^{in}_{h,2}) + U_2 A_2(T^{in}_{h,2} - T_2 + T_{h,2} - T_0) \end{bmatrix} \tag{4.14}$$

The measured variables are highlighted in red in Figure 4.2, the rest are unknowns which must be eliminated. Following the same procedure as in Chapter 4.1 (the `Maple` code can be found in Appendix H.2) the controlled variable is:

$$c = \left(\frac{T^{in}_{h,1} - T_1}{T^{in}_{h,11}} - 1 \right) \frac{T^2_{11}}{T^{in}_{h,1} - T_{11}} + \frac{T^2_1}{T^{in}_{h,1} - T_{11}} - \frac{T^2_2}{T^{in}_{h,2}} = 0 \tag{4.15}$$

Also here the temperatures are "Δ"-temperatures, i.e. should be subtracted T_0.

Equation (4.15) reduces to Equation (4.13) when the area of HX1 goes to zero, i.e. $T_1 = T_{11}$:

$$\begin{gathered} c = \left(\frac{T^{in}_{h,1} - T_{11}}{T^{in}_{h,11}} - 1 \right) \frac{T^2_{11}}{T^{in}_{h,1} - T_{11}} + \frac{T^2_{11}}{T^{in}_{h,1} - T_{11}} - \frac{T^2_2}{T^{in}_{h,2}} = 0 \\ c = \left(\frac{T^{in}_{h,1} - T_{11}}{T^{in}_{h,11}} \right) \frac{T^2_{11}}{T^{in}_{h,1} - T_{11}} - \frac{T^2_2}{T^{in}_{h,2}} = 0 \\ c = \frac{T^2_{11}}{T^{in}_{h,11}} - \frac{T^2_2}{T^{in}_{h,2}} = 0 \end{gathered} \tag{4.16}$$

where HX11 is equivalent to HX1 in the previous case.

Note that we can divide the controlled variable in the following manner:

$$f_1 = \left(\frac{T^{in}_{h,1} - T_{11}}{T^{in}_{h,11}} - 1 \right) \frac{T^2_{11}}{T^{in}_{h,1} - T_{11}} + \frac{T^2_{11}}{T^{in}_{h,1} - T_{11}} \tag{4.17}$$

$$f_2 = \frac{T^2_2}{T^{in}_{h,2}} \tag{4.18}$$

Then the controlled variable is $c = f_1 - f_2$. With an approach like this it is possible to consider two lines at a time and the self-optimizing control strategy proposed can be extended to multiple streams.

5 Case Studies

Now the self-optimizing variables found in Chapter 4 will be investigated through two case studies: First with two heat exchangers in parallel and then with two heat exchangers in series and one in parallel. For each case study four variables are plotted against the split u: The end temperature T_{end}, the controlled variable c, θ_1/θ_2 and the error related to the approximation $\Delta T_{LM} \approx \Delta T_{AM}$.

The plot of the controlled variable c indicates which split a self-optimizing control strategy results in, i.e. where the curve intersects 0 on the y-axis. The vertex of the T_{end} plot indicates where the optimal split is located. The self-optimizing split should be close to the optimal split, i.e. the "loss" should be acceptable.

For both the case studies the inlet temperature T_0 is 60°C and the cold inlet heat capacity rate ($w_0 = mC_{p,0}$) is 100 W/K, while the heat exchanger UA-values and hot heat capacity rates and inlet temperatures vary. First, the hot and cold outlet temperatures are calculated for every split using the ϵ-NTU method from Chapter 3.1.4. Assuming ideal mixing the end temperature is given by $T_{end} = uT_1 + (1-u)T_2$. When all the cold outlet temperatures are known the controlled variable can be calculated. Note that we use the self-optimizing strategy derived from the arithmetic mean temperature difference approximation to control a HEN where the heat transfer is driven by the logarithmic mean temperature difference.

The `MATLAB` software has been used to do the calculations, and the m-files can be found in Appendix G.1.

5.1 Case I: Two Heat Exchangers in Parallel

With two heat exchangers in parallell the controlled variable is given in Equation (4.13) and the the flowsheet in Figure 4.1. One case study will be presented here while six other case studies are presented in Appendix A.

The parameters for Case I-a are given in Table 5.1.

The temperature as a function of the split and the self-optimizing variable as a function of the split is shown in Figure 5.1. By keeping c constant and equal to zero, maximum end temperature is achieved ($T_{end} \approx 98.1$°C). The optimal split and

	ω_h [W/K]	$T_{h,in}$ [°C]	UA [W/K]
HX1	30	120	50
HX2	50	140	80

Table 5.1: Parameters for Case I-a

the self-optimizing split are very close, 0.35 and 0.34, respectively. The temperature profiles for both the optimal split and the self-optimizing split are shown in Figure 5.2.

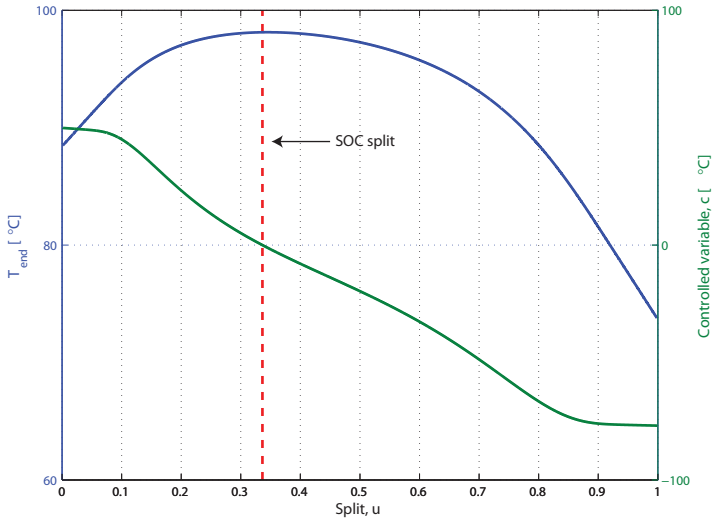
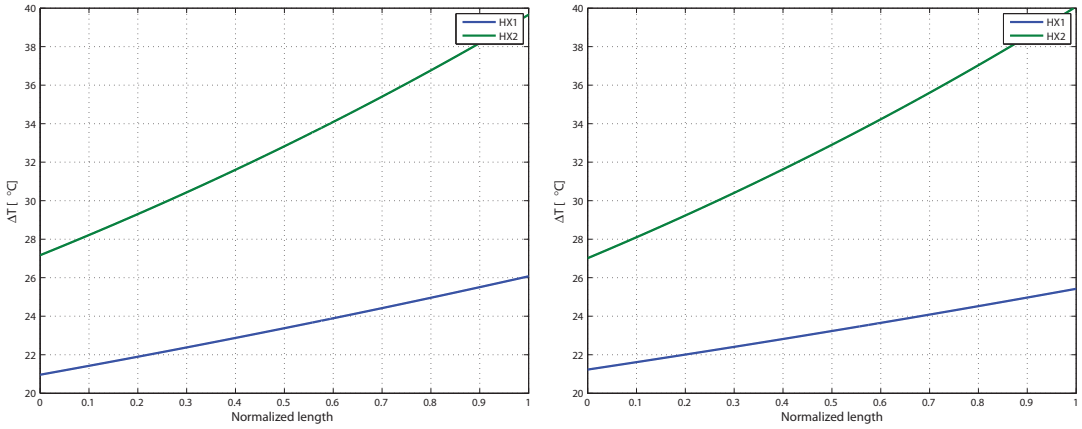


Figure 5.1: T and c vs split

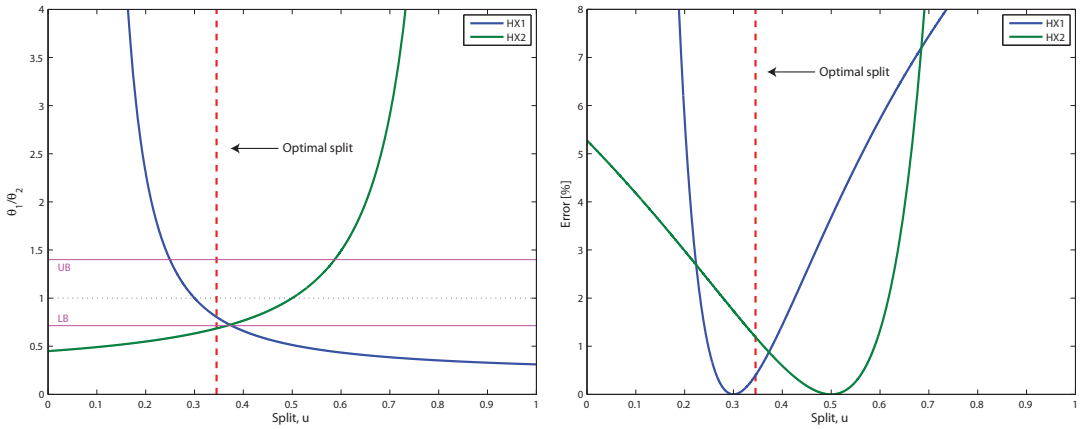
To check how good the approximation $\Delta T_{LM} \approx \Delta T_{AM}$ is for the given case a plot of θ_1/θ_2 vs the split is made in Figure 5.3(a). If $1/1.4 < \theta_1/\theta_2 < 1.4$ (marked with pink lines as LB and UB) for both HX1 and HX2 the error will be less than 1% according to Skogestad (2003a). In Figure 5.3(b) the actual error is shown. The low point for the curves (Error = 0%) indicates where $\theta_1/\theta_2 = 1$.

The resulting end temperature when using self-optimizing control is close to the optimal end temperature for all the seven cases, even when the approximation error is relatively large.



(a) Optimal (b) Self-optimizing

Figure 5.2: Temperature profiles



(a) Approximation (b) Error

Figure 5.3: Approximation

5.2 Case II: Two Heat Exchangers in Series and One in Parallell

With two heat exchangers in series and one in parallell the controlled variable is given in Equation (4.15) and the the flowsheet in Figure 4.2. One case study will be presented here while five other case studies are presented in Appendix B.

The parameters for Case I-b are given in Table 5.2.

	ω_h [W/K]	$T_{h,in}$ [°C]	UA [W/K]
HX11	30	120	50
HX1	50	140	140
HX2	20	140	65

Table 5.2: Parameters for Case I-b

The temperature as a function of the split and the self-optimizing variable as a function of the split is shown in Figure 5.4. Again, by keeping c constant and equal to zero, maximum end temperature is achieved ($T_{end} \approx 107.6$°C).

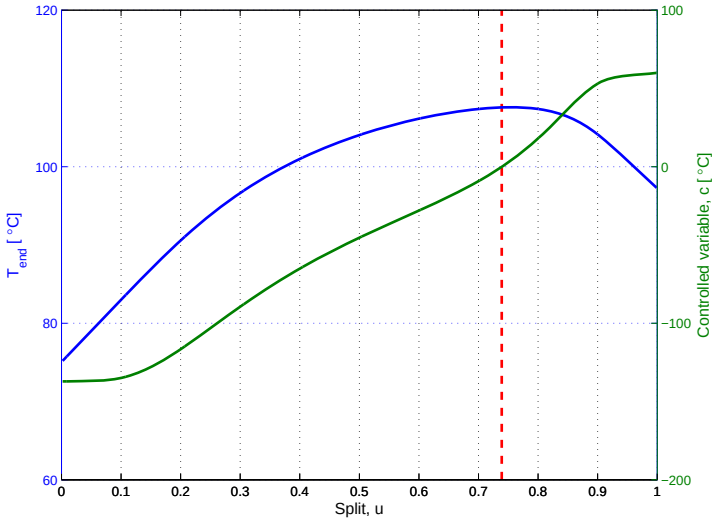
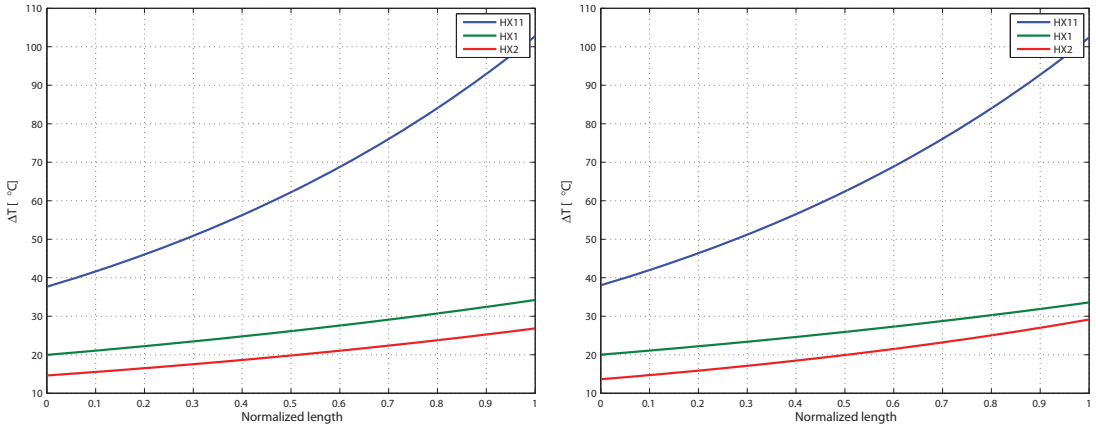


Figure 5.4: T and c vs split

The temperature profiles for both the optimal split and the self-optimizing split are shown in Figure 5.5.

To check how good the approximation $\Delta T_{LM} \approx \Delta T_{AM}$ is for the given case a plot of θ_1/θ_2 vs the split is made in Figure 5.6(a). If $1/1.4 < \theta_1/\theta 2 < 1.4$ for both HX1 and HX2 the error will be less than 1% according to (Skogestad, 2003a). In Figure 5.6(b) the actual error is calculated. The low point for the curves (Error = 0%) indicates where $\theta_1/\theta_2 = 1$.

In Case B.3 (see Appendix B) T_{soc} is deviating a bit from T_{opt} (118.5°C vs 120.5°C, respectively). The reason for this is the high error assosciated with approximating

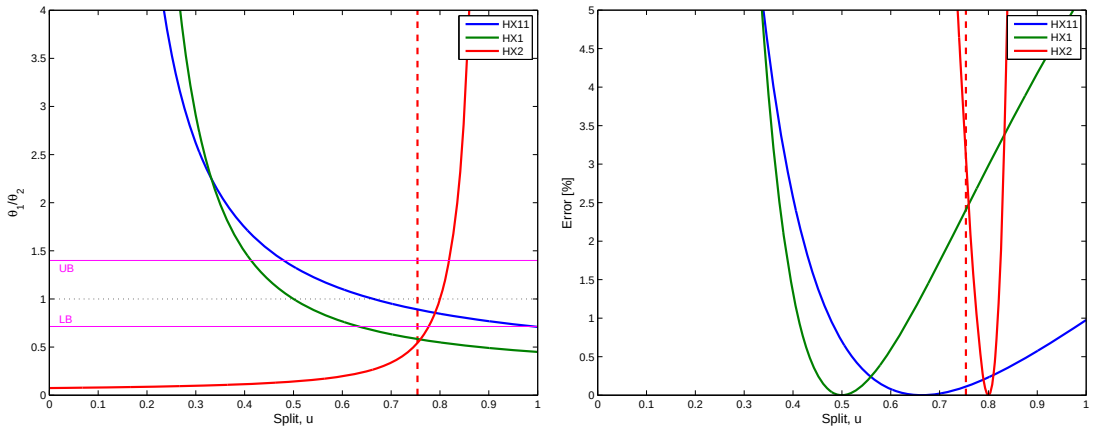


(a) Optimal (b) Self-optimizing

Figure 5.5: Temperature profiles

LMTD with the arithmetic mean temperature difference, illustrated in Figure B.9. θ_1/θ_2 is approximately 5 for HX2 resulting in an error of about 20%. In the other cases the self-optimizing end temperature is close to the optimal end temperature.

Also, by looking on Case B.5 there does not exist any solution for this case. According to Figure B.15 the error assosciated with the approximation is about 25% for HX2. The reason is because θ_1 is about six times larger than θ_2 for HX2.



(a) Approximation (b) Error

Figure 5.6: Approximation

6 Perstorp Study

A case study has been performed on a heat exchanger network for a company producing a wide range of chemicals called Perstorp, located in Perstorp, Sweden. The heat exchanger network is shown in Figure 6.1. The actual flowsheet is presented Figure C.1.

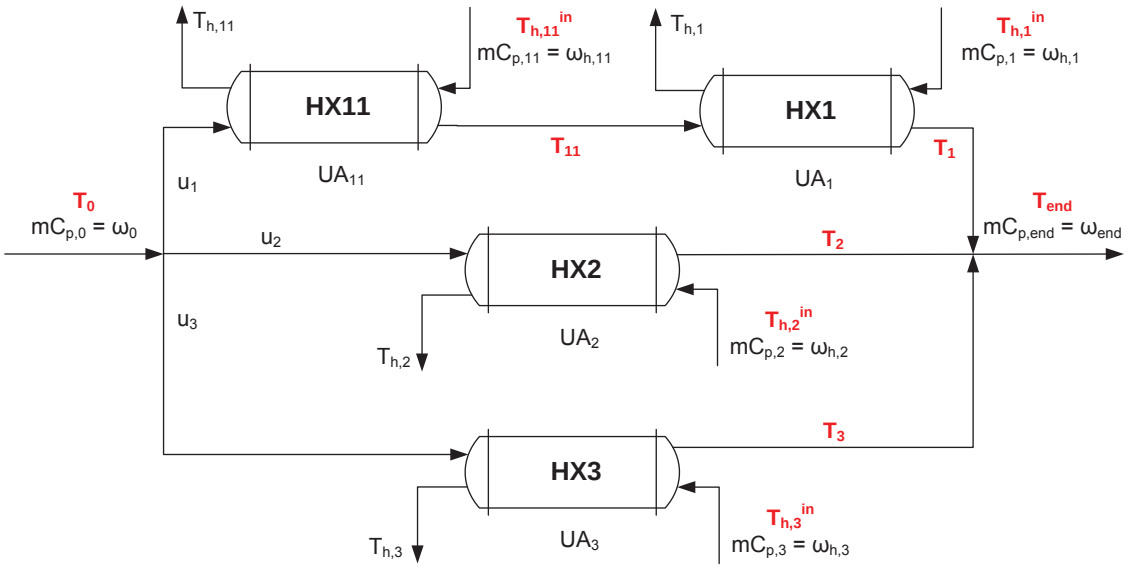


Figure 6.1: Simplified flowsheet for the Perstorp HEN

As can be seen from Figure 6.1 the heat exchanger network is build up of two cases already studied. HX2 and HX3 constitute Case I while HX11, HX1 and HX2 (or HX3) constitute Case II, which means we can use the self-optimizing controlled variables already found. We want to compare two and two "networks"; this can be done since it will assure that all the controlled variables are zero. Flow measurements at Perstorp reveals that stream 2 is the largest stream (in other words, u_2 is the largest mass fraction). Thus, the temperatures related to HX2 will be present in both the control loops. This will lead to the best dynamic performance. Defining

$$f_1 = \left(\frac{T_{h,1}^{in} - T_1}{T_{h,11}^{in}} - 1 \right) \frac{T_{11}^2}{T_{h,1}^{in} - T_{11}} + \frac{T_1^2}{T_{h,1}^{in} - T_{11}} \tag{6.1}$$

$$f_2 = \frac{T_2^2}{T_{h,2}^{in}} \tag{6.2}$$

$$f_3 = \frac{T_3^2}{T_{h,3}^{in}} - \frac{T_2^2}{T_{h,2}^{in}} \tag{6.3}$$

the controlled variables are

$$c_1 = f_1 - f_2 \tag{6.4}$$

$$c_2 = f_3 - f_2 \tag{6.5}$$

where all the temperatures are "Δ"-temperatures, i.e. should be subtracted the inlet temperature T_0. The setpoints for the controlled variables are $c_{1s} = c_{2s} = 0$.

In Figure 6.2 the suggested control structure is presented, with two control loops present. Since valve dynamics is not taken into consideration it is assumed that flow controllers are used. To achieve feed back control the two flows in stream 1 and stream 3 must be measured (F_m). There exists already one flow measurement on stream 1 (`FT2047` in Figure C.1) which means that there should be installed a flow transmitter on stream 3 as well. Alternatively the valve opening could be set directly by using a valve positioner which eliminates the need for flow measurements.

All the measured variables (temperatures) are highlighted in red. Potential disturbances in the HEN are all the hot inlet streams ($T_{h,11}^{in}$, $T_{h,1}^{in}$, $T_{h,2}^{in}$ and $T_{h,3}^{in}$). If, for example, $T_{h,3}^{in}$ suddenly increases, then c_2 from Equation (6.5) will be negative. To compensate for this the valve opening is increased, T_3 is the decreased and c_2 will eventually go back to zero. The inlet temperature T_0 could also be considered a disturbance.

6.1 Assumptions

Several assumptions are made to simplify the problem:

- As in all cases, the mC_p is assumed to be constant throughout the heat exchanger. The inlet mC_p (hot and cold) is used.
- Pressure and valve position have not been taken into consideration. Thus, it is assumed that flow controllers are used to set the flow in the different streams.

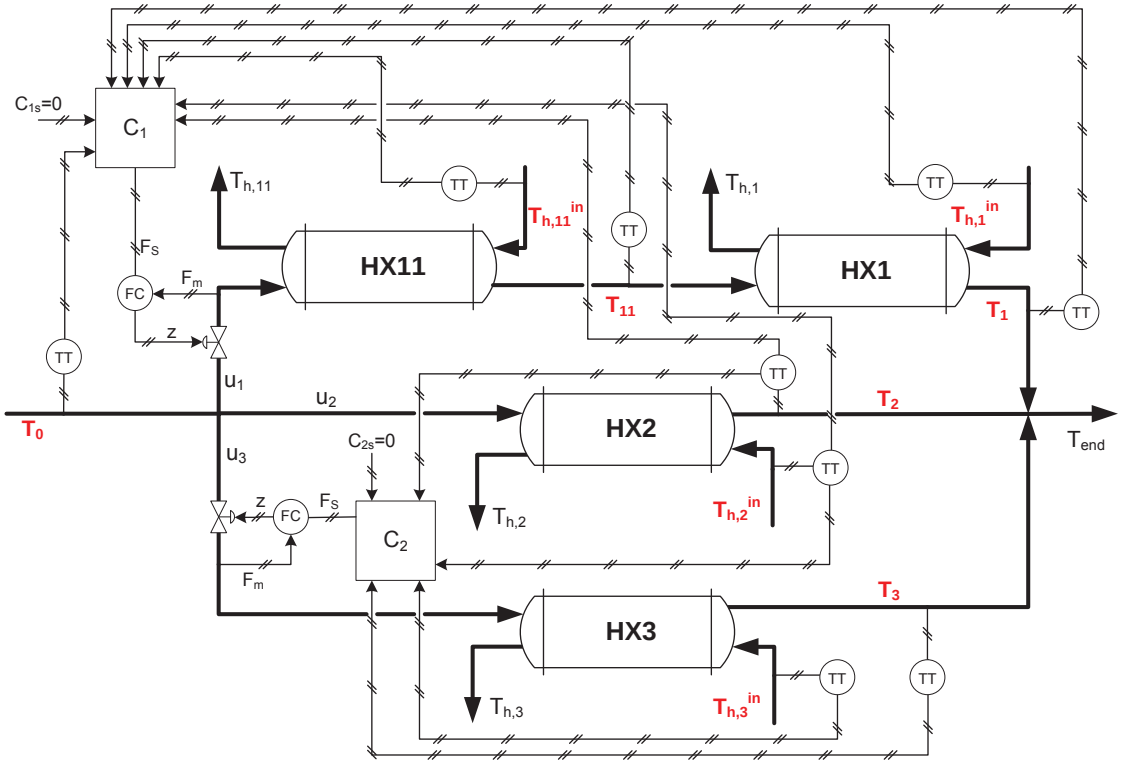


Figure 6.2: Control structure for Perstorp HEN

- Ideal mixing is assumed, i.e. $T_{end} = u_1 T_1 + u_2 T_2 + u_3 T_3$.
- There exists a bypass on HX11 (H-202 in Figure C.1). This bypass was taken into consideration through the heat transfer coefficient for HX11. It is assumed that the total flow (the flow which actually flows through HX11 plus the bypass flow) goes through HX11. Then the heat transfer coefficient is lowered to match the inlet temperature of HX1 measured at Perstorp. Note: It is the product $h \times A$ which matters (see Equation (3.28), (3.29) and (3.30)). I.e., either h or A could be adjusted.
- Detailed specifications was only available for HX1 (wall density and wall heat capacity). Wall densities and heat capacities for the other heat exchangers are assumed to be the same. It is assumed that the flow pattern is ideal countercurrent for all the heat exchangers. This will not correspond to the real case as HX1 is a plate and fin heat exchangers and the rest are spiral plate heat exchangers.
- The volume was not available for HX1, but it was for another plate and fin heat

exchanger. The volume for HX1 is assumed to follow the same area-volume relationship as for this heat exchanger.

- The heat transfer coefficients has been estimated based on only one data point ($k = 4000$) and it is assumed that the heat tranfer coefficients are the same for the other data points as well. The heat transfer coefficients were adjusted until outlet temperatures corresponded with measured temperatures from Perstorp.

6.2 Steady-State Analysis

A steady-state analysis of the Perstorp HEN has been conducted using the `MATLAB` software. The m-file '`perstorp_ss.m`' can be found in Appendix G.2. Relevant data for three years (2008, 2009 and 2010) was received from the Perstorp company and has been used in the simulations.

From Figure C.1 it can be seen that there does not exist flow transmitters on the cold streams, except on the upper cold stream. Therefore, energy balances was used to estimate the UA values for the heat exchangers. C_p values were received from Perstorp and Equation (3.6) could be used to estimate the UA values since all the temperatures are known.

Using the suggested control strategy presented above the end temperature using self-optimizing control can be found and compared to the optimal end temperature. Also, the actual end temperature is measured at Perstorp. In summray, three temperatures are compared: The optimal end temperature (T_{opt}), the end temperature using self-optimizing control (T_{soc}) and the actual end temperature measured at Perstorp (T_{m}).

The result for one of the data points in 2010 (July 30, 6pm) is presented in Figure 6.3. The plot indicates how large the loss is when self-optimizing control is used. Ideally, $T_{\text{opt}} = T_{\text{soc}}$, but this will never be the case because of the approximation mentioned in Chapter 4.1 and Chapter 4.2 and disturbances, but we should be sure that the loss is acceptable. The thinner red contour lines in the figure indicates which end temperature (in °C) the different splits result in, and in the middle of these contour lines the location of T_{opt} is plotted (black dot). Further, the thicker blue line indicates where the self-optimizing control variables $c_1 = 0$ and the thicker red line indicates where the self-optimizing control variables $c_2 = 0$. The resulting

temperature by using self-optimizing control, T_{soc} is when both of the controlled variables are zero, i.e. where the lines intersect. The end temperature measured at Perstorp (T_{p}) is also indicated in the figure, but the exact splits are unknown because of inconsistent flow measurements. Thus, T_{m} is plotted as a contour line as well. The end temperatures are indicated with labels with yellow background and T_{m} with green background.

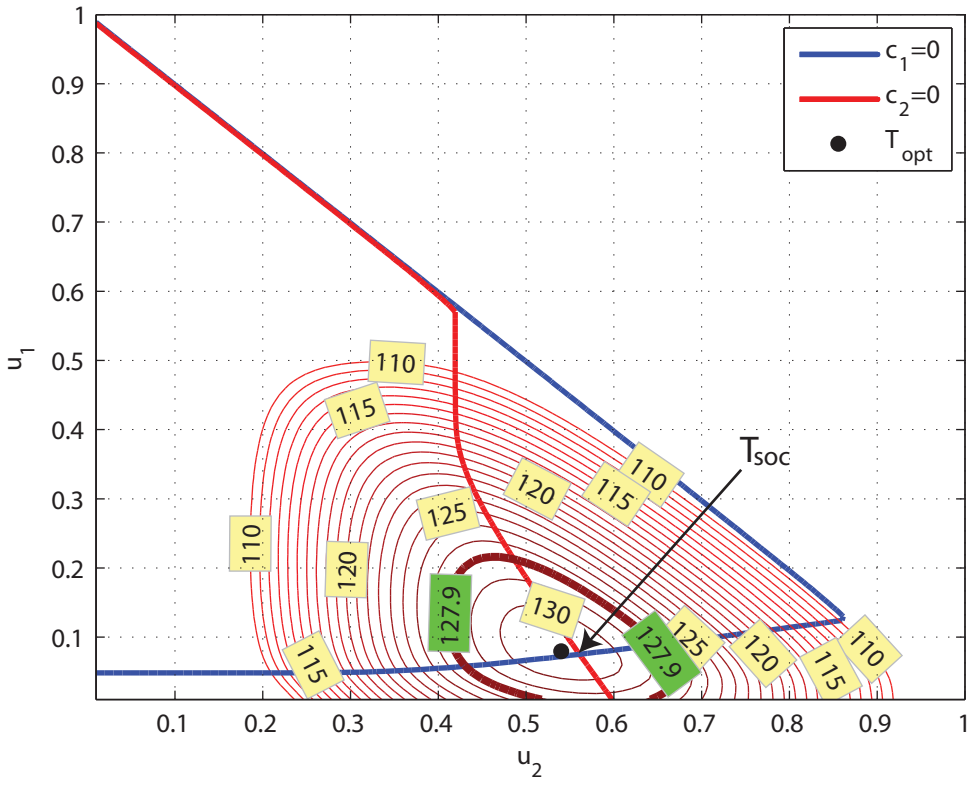


Figure 6.3: Steady-state results (July 30, 6pm)

From Figure 6.3 it is easy to see that T_{soc} is close to $T_{\mathrm{opt}} = 130.8°\mathrm{C}$. Several other case studies are presented in Appendix D. From the figures it can be seen that T_{soc} is close to T_{opt} in all the cases.

It is only the lower left triangle of the plane Span$\{u_1, u_2\}$ that is valid since $u_1 + u_2 + u_3 = 1$. Since u_3 is in the region $0 \leq u_3 \leq 1$, then also $0 \leq u_1 + u_2 \leq 1$. The potential for improvement is investigated by plotting the difference $T_{\mathrm{opt}} - T_{\mathrm{m}}$ for the whole year. These results are presented in Figure 6.4, Figure 6.5, Figure 6.6 for the years 2008, 2009 and 2010, respectively.

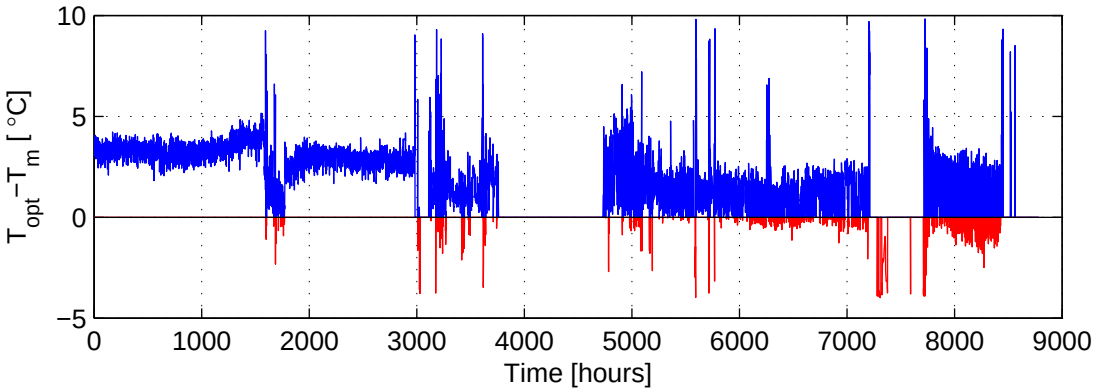


Figure 6.4: Yearly variations in 2008

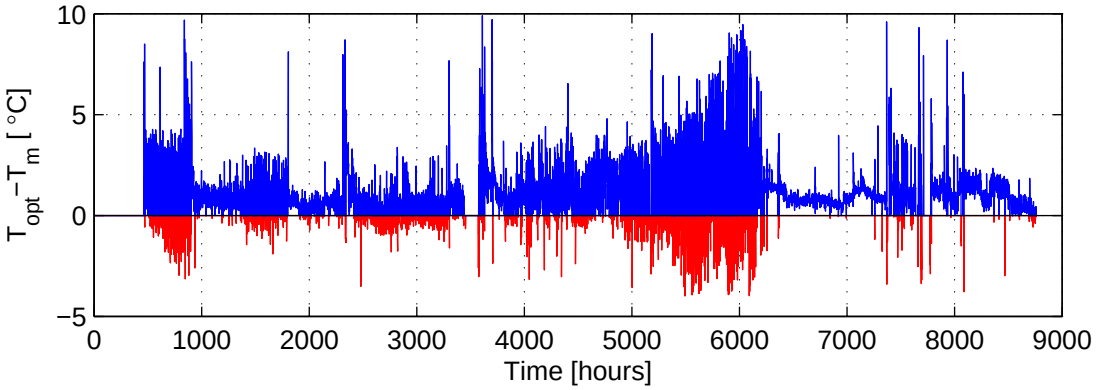


Figure 6.5: Yearly variations in 2009

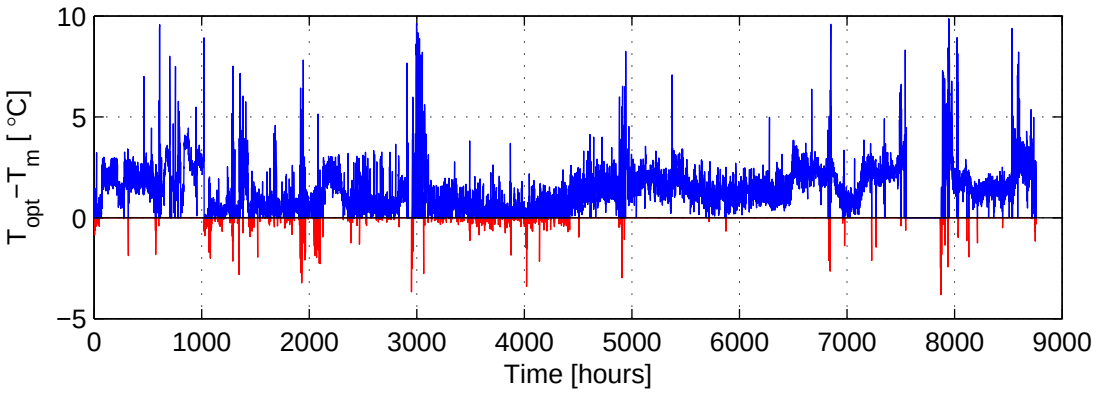


Figure 6.6: Yearly variations in 2010

From the three figures it can be seen that the potential for increasing the end temprature is high. Especially in the beginning of 2008 the potential for improvement was high (about 3.5°C), but also in the end of 2010 there was some potential.

Some of the temperature differences are negative (marked with red); this indicated that unaccurate parameters have been used in the simulations, for instance the heat capacity. Another reason can be that the operation is at non-steady state. Very high temperature differences also indicate non-steady state operation. The potential for improvement seems to be about $1 - 3$°C.

6.3 Dynamic Model

Using the equations presented in Chapter 3.1.5 the heat exchangers were modelled using the `Simulink` software. The flowsheet is presented in Figure 6.7. Heat exchanger volumes and areas together with the heat exchanger wall mass and density were received from Perstorp. The heat transfer coefficients were estimated by simulations; they were either increased or decreased to match the outlet temperature out of each heat exchanger measured at Perstorp.

The model order of the dynamic model is 10 which will assure good accuracy. The number of initial conditions in the dynamic model is 122. 30 for each heat exchanger (10 for hot side, 10 for the wall and 10 for cold side) and 1 for each controller. Since there seemed to be normal operation around hour 4000 in 2010 (June 15, 4pm) this point was used when the heat transfer coefficients were estimated. Thus, all the initial values for the four heat exchangers and the two controllers were taken from steady-state at this point. The inital values were implemented as matrices, which means that when other datas are used for simulation the system will not start at steady state. The Simulink file was run from `Perstorp_dynamic.m`, see Appendix G.2.

The controllers are tuned using the Skogestad IMC (SIMC) PID tuning rules (Skogestad, 2003b). Controller loop 1 is tuned first. After tuning the loop is closed and controller loop 2 is tuned. The controllers are tuned by making a step (10% increase) in the mass flows in the respective streams. The step and responses are presented in Figure 6.8. The resulting PID tuning parameters for Controller 1 are presented in Table 6.1.

Now some disturbances are applied. First, a temperature increase of 1% (from 153.2 °C to 154.7°C) in $T_{h,2}^{in}$ is applied. The responses of the controlled variables

	$k_c[\frac{^\circ C}{kg/s}]$	τ_I [s]
Controller 1	0.026	62
Controller 2	0.102	174

Table 6.1: PI tuning parameters

are shown in Figure 6.9. From the figure an inverse response is observed when the disturbance kick in at t = 2min. The controller output is given by Equation (6.6).

$$c(t) = \bar{c} + K_c e(t) + \frac{1}{\tau_I} \int_0^t e(t^*)\, \mathrm{d}t^* \tag{6.6}$$

where $\bar{c}$ is the steady-state value of the controller, e is the error signal to the controller, K_c the controller gain and τ_I the integral time. The error signal e is the same as the controlled variable in this case since the setpoint is zero. I.e., when $T^{in}_{h,2}$ is increased both c_1 and c_2 (by looking on Equation (6.4) and Equation (6.5)) will turn positive immediately. The error signal is not zero (inverse response) and the controller tries to counteract this. Inverse responses are associated with right-half plane zeros (Seborg et al., 2003) and arrise from competing dynamic effects that operate on two different time scales (in this case since we have two control loops with interaction). The inverse response could be lowered by implementing a filter on the disturbance, but this has not been done in this study. The undershoot observed in Figure 6.9 could be also lowered by decreasing the controller gain. The time to reach setpoint ($c_{1s} = c_{2s} = 0$) is about 15 minutes, which could be considered a fast response since temperature changes are slow processes.

To easier visualize the effect of a hot temperature increase the temperature raise of $T^{in}_{h,2}$ is increased to 10%. The plot of how the splits change is presented in Figure 6.10 and the cold outlet temperatures in Figure 6.11. u_2 has increased from 0.570 to 0.597 while u_1 and u_3 have decreased from 0.047 to 0.044 and 0.383 to 0.359, respectively. The end temperature T_{end} has increased from 126.9°C to 133.4°C. Split and temperature responses for a 10 % increase in $T^{in}_{h,3}$ is shown in Figure 6.12 and Figure 6.13, respectively. Since the mass flow through stream 1 is relatively low compared to stream 2 and 3, a temperature increase of 10 % for $T^{in}_{h,11}$ or $T^{in}_{h,1}$ does not change the splits or cold temperatures out very much, hence the plots are not shown here.

By using the proposed self-optimizing control strategy it is assured that the controlled variables reaches their sepotins and good disturbance rejection is also achieved.

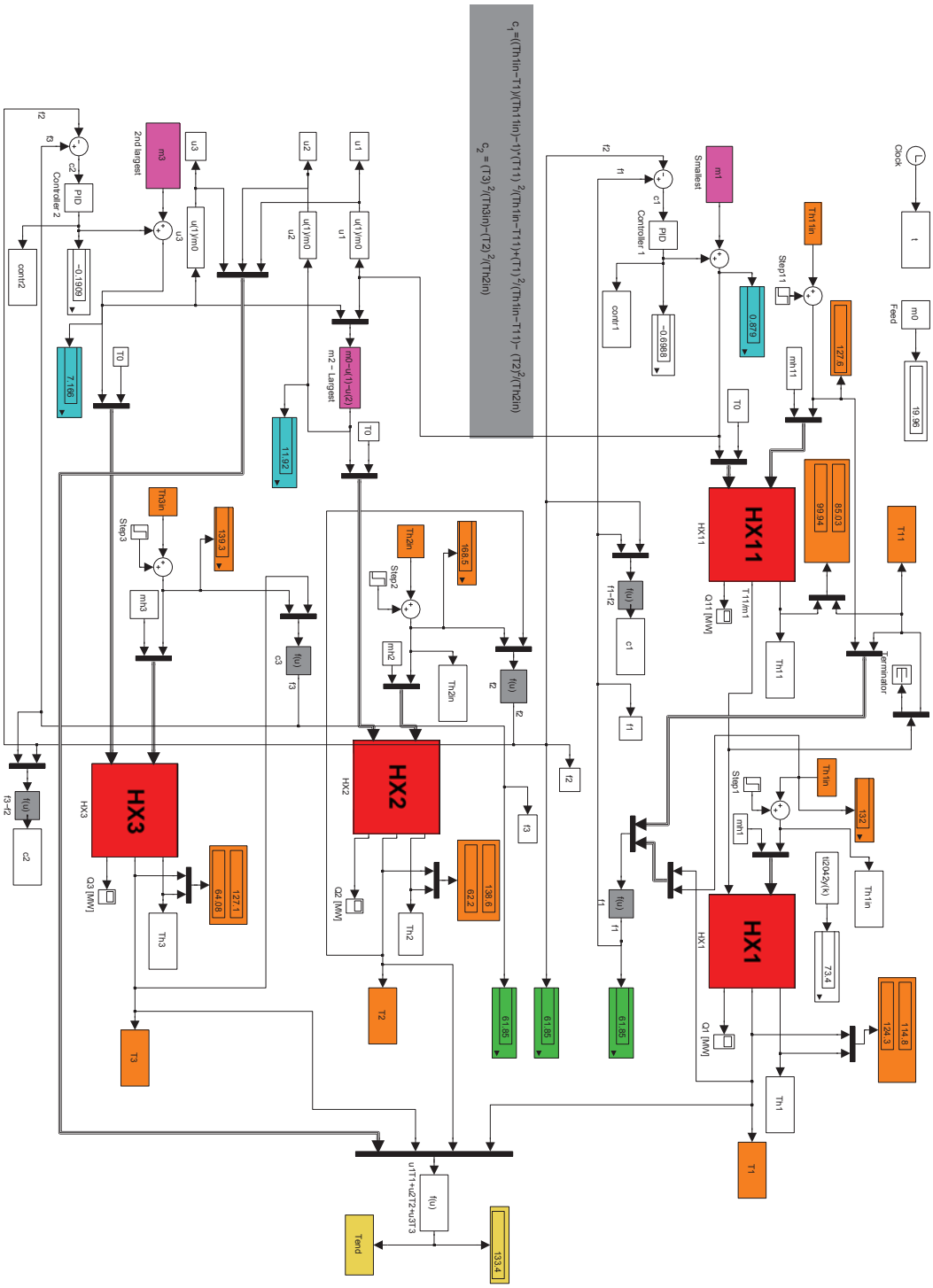
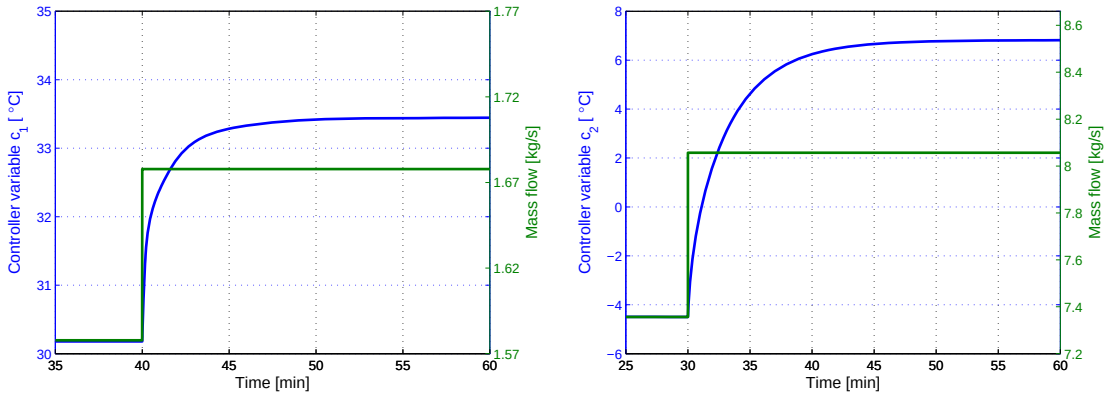


Figure 6.7: Flowsheet of Perstorp HEN in Simulink



(a) c_1 (b) c_2

Figure 6.8: Step responses

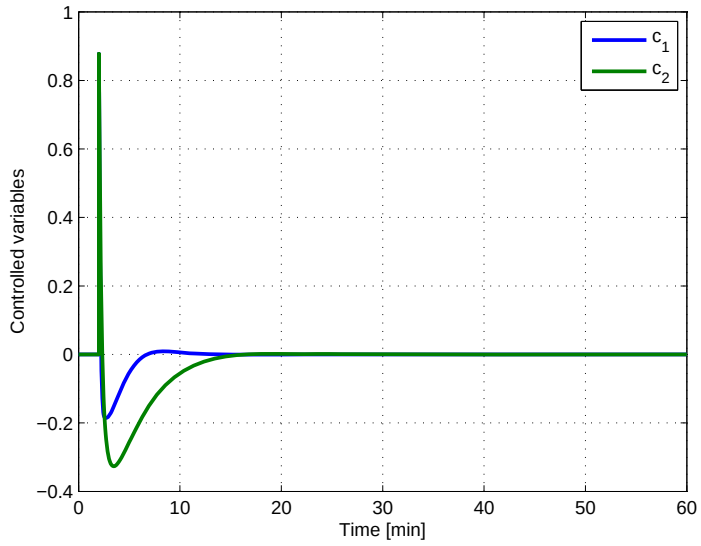


Figure 6.9: Response of controlled variables when $T_{h,2}^{in}$ is increased

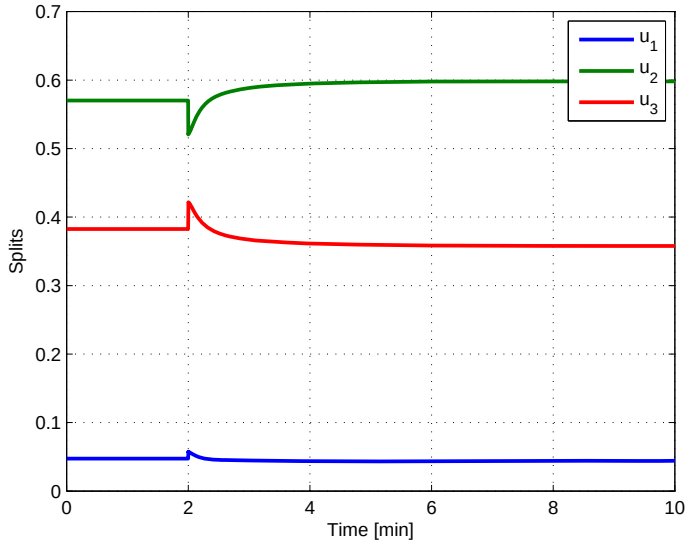


Figure 6.10: Response of splits when $T_{h,2}^{in}$ is increased by 10 %

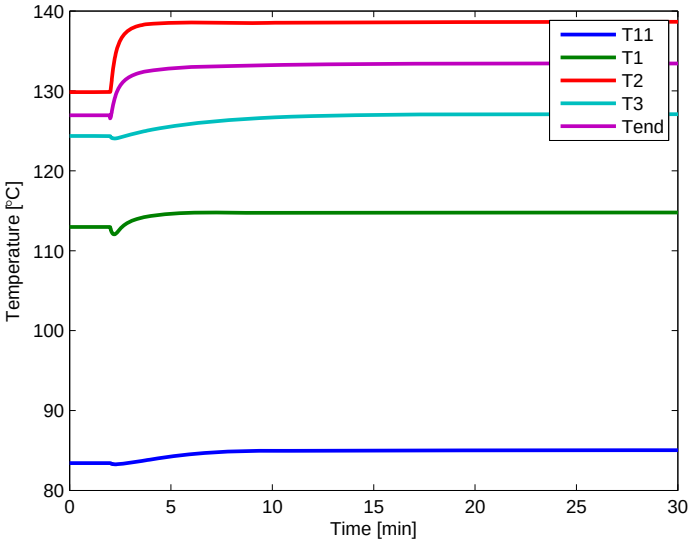


Figure 6.11: Response of $T_{c,out}$ and T_{end} when $T_{h,2}^{in}$ is increased by 10 %

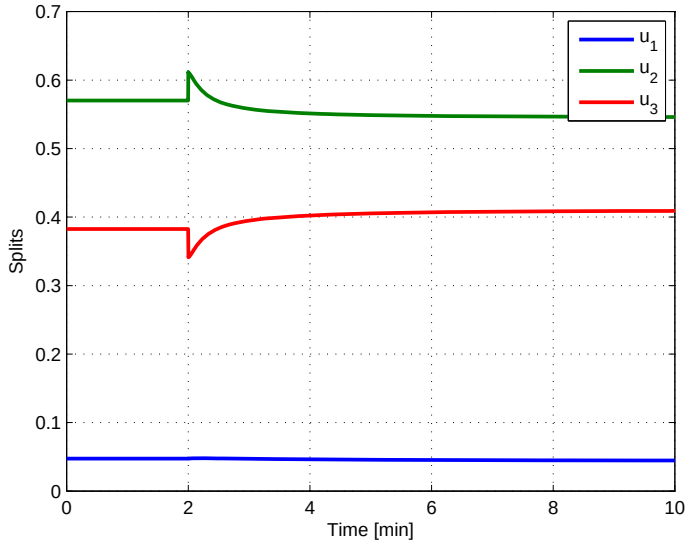


Figure 6.12: Response of splits when $T_{h,3}^{in}$ is increased by 10 %

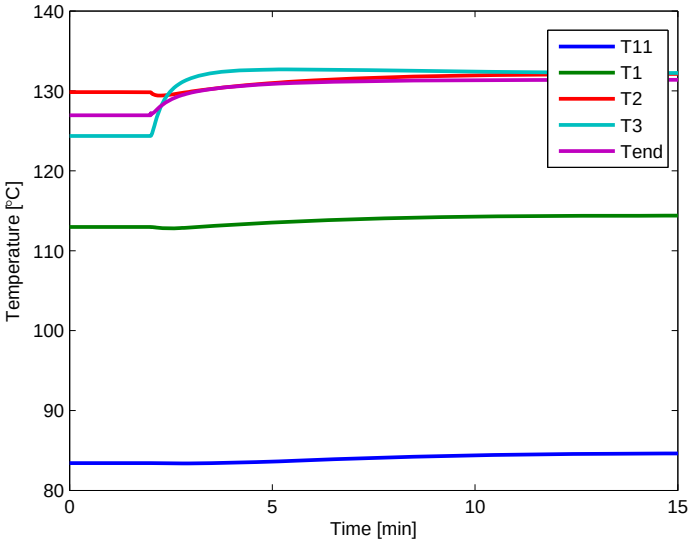


Figure 6.13: Response of $T_{c,out}$ and T_{end} when $T_{h,3}^{in}$ is increased by 10 %

7 Crude Unit Heat Exchanger Network

7.1 Introduction

As mentioned earlier preheating of crude oil in oil refineries is also a place where the optimization objective is to save energy by recovering as much heat as possible, i.e. maximize the end temperature. A typical heat exchanger network is shown in Figure 7.1 (Lid and Skogestad, 2001).

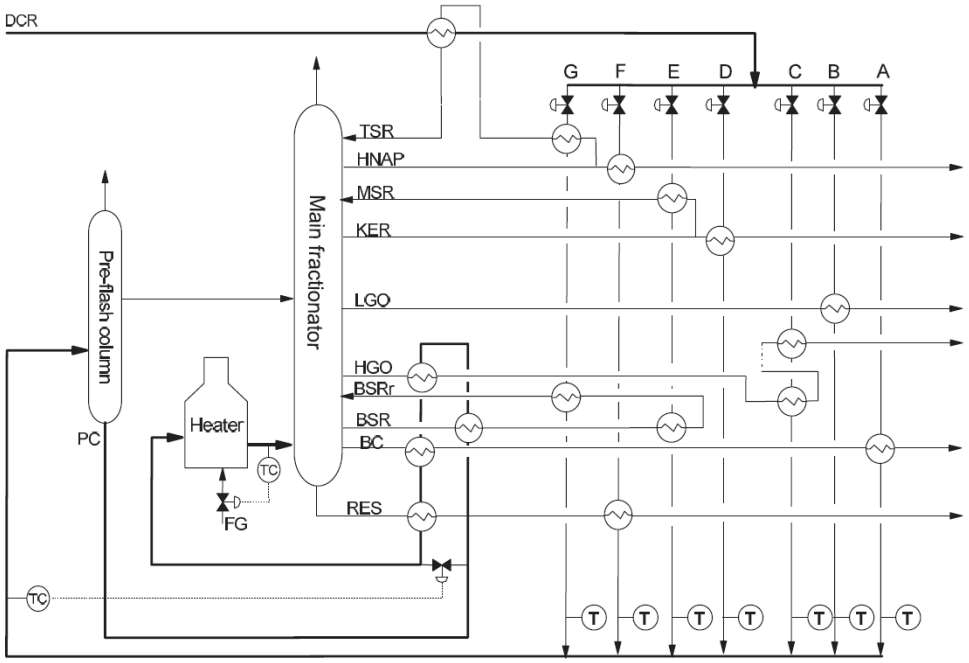


Figure 7.1: Typical HEN for preheating of crude oil

In this case the cold crude oil is split in seven parallell streams (A-G), i.e. six splits, to be preheated by various hot streams from the main fractionator. A study has been performed for Statoil Mongstad which is the largest refinery in Norway with a capacity of 10 million tonnes crude oil per year (Statoil, 2011). The crude

unit HEN is controlled using RTO and this study will investigate if the proposed self-optimizing control strategy can be used and improve the performance.

A simplified flowsheet of the heat exchanger network at Statoil Mongstad is presented in Figure 7.2. The number after the stream name (86.7 for stream A etc.) are flows in tonnes/h. The blue numbers are cold temperatures and the red are hot temperatures in °C. The actual heat exchanger network is shown in Appendix E.

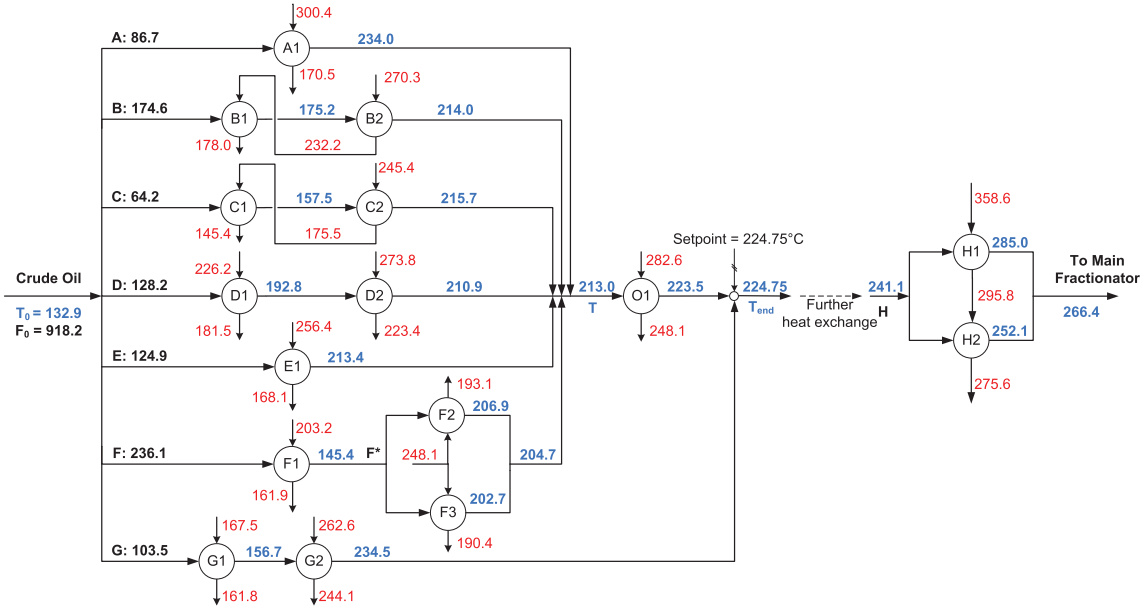


Figure 7.2: Simplified heat exchanger network at Mongstad

All the streams are in complete liquid phase, according to Statoil Mongstad. Hence, no evaporation occurs which justfies the constant mC_p assumption. To calculate the outlet temperatures for the heat exchangers B1, B2, C1 and C2 an iterative approach has to be used since B1 and C1 cannot be solved before B2 and C2 are solved, respectively. The `MATLAB` function *fsolve* has been used for this purpose. Given an inital hot inlet temperature $T_{h,1}^{in}$ (for B1 and C1) the function finds the hot outlet temperature $T_{h,2}$ (for B2 and C2) that matches $T_{h,1}^{in}$. The `MATLAB` code is given in Appendix G.3.

The simplified flowsheet shows that there are maximum two heat exchangers in series on each stream. At first sight one could think that the HEN could be controlled in the same manner as demonstrated earlier. However, since stream G is not united with the other streams until after the heat exchanger O1 it is not

as straight forward. One of the main advantages with the self-optimizing method proposed in this study is that not the entire heat exchanger network needs to be included. Thus, letting stream G remain constant at a flow rate of 103.5 tonnes/h, the rest of the HEN could be optimized.

Data (temperatures, UA-values, heat capacities and flows) have been received from Statoil Mongstad and it is realized that stream F is the largest stream with a flow rate of about 236 tonnes/h. Thus, this stream should be used as the reference stream (as stream 2 was used as reference stream in the Perstorp Case). This is only for dynamic purposes; the controlled variables will be zero regardless of which stream is selected as reference stream, hence it will not affect the steady state operation. Stream F is a bit special since it involves a heat exchanger *prior* to a split (F1) and two succeeding heat exchanger in parallell (F2 and F3). Following the same procedure as in Chapter 4.1 and Chapter 4.2 the self-optimizing variable could be found for this case as well, but it would most likely be complicated. However, since both F2 and F3 have the same hot inlet stream these two heat exchangers can be simulated as one heat exchanger. The new heat exchanger, denoted F*, will then have $UA_{F^*} = UA_{F2} + UA_{F3}$ with heat capacity rate $\omega_{F^*} = \omega_{F2} + \omega_{F3}$ and stream F is simulated with two heat exchangers in series.

It is worth checking how good the arithmetic mean temperature difference approximation is. In this case shell-and-tube heat exchangers are used which means that the correction factor F must be included. Thus, it is investigated how large the error assosciated with the approximation $\Delta T_{LM} F \approx \Delta T_{AM}$ is. The result is presented in Table 7.1 along with hot and cold heat capacity rates, the ratio of the temperature differences on both sides of each heat exchanger, number of shells, LMTD, correction factor and AMTD. The inlet and outlet temperatures for each heat exchanger can be found in Figure 7.2.

From the table it can be seen that there is a large error associated with some of the heat exchangers, espescially A1, C1, C2, D2, F1, G1 and G2. Either the hot and cold heat capacity rates for these heat exchangers differ in magnitues or the correction factor is low. For C1 and C2 the correction factor is 0.84 and 0.85, respectively, resulting in a high error. For G1 and G2 the hot and cold heat capacity rates are very different (331 % and 356 %, respectively) resulting in a high error. These relatively large errors may contribute to that the self-optimzing control strategy will

	$mC_{p,c}\left[\frac{\text{kW}}{\text{K}}\right]$	$mC_{p,h}\left[\frac{\text{kW}}{\text{K}}\right]$	θ_1/θ_2	N_{shells}	$\Delta T_{LM}[°\text{C}]$	F	$\Delta T_{LM}F[°\text{C}]$	$\Delta T_{AM}[°\text{C}]$	Error [%]
A1	56.74	52.76	1.77	4	50.62	0.94	47.79	51.98	8.78
B1	114.24	125.91	1.26	2	50.82	0.96	48.87	51.05	4.45
B2	122.98	101.95	0.99	2	56.63	0.98	55.53	56.64	1.98
C1	42.02	36.70	1.39	2	14.81	0.84	12.47	14.94	19.81
C2	43.89	40.80	1.59	3	23.69	0.85	20.10	24.12	19.99
D1	83.90	122.84	0.69	4	40.54	0.98	39.84	41.01	2.93
D2	92.98	351.83	2.05	1	44.88	0.92	41.15	46.80	13.74
E1	81.81	85.61	1.22	4	38.94	0.95	36.96	39.07	5.72
F1	154.48	49.05	2.00	1	41.72	0.95	39.51	43.37	9.76
F2	74.80	91.65	0.86	4	44.34	0.98	43.53	44.42	2.04
F3	83.17	90.37	1.01	4	45.21	0.98	44.43	45.21	1.74
G1	67.69	291.48	0.37	1	18.40	0.93	17.04	19.86	16.58
G2	70.60	321.75	0.32	1	52.25	0.90	46.90	57.76	23.15
H1	217.13	159.61	1.32	2	64.22	0.97	62.41	64.63	3.56
H2	289.54	147.97	1.29	2	39.35	0.99	39.10	39.56	1.19
O1	610.30	190.55	1.69	1	46.02	0.97	44.66	47.07	5.38

Table 7.1: Error associated with the approximation $\Delta T_{LM}F \approx \Delta T_{AM}$

deviate from the optimal point.

To make sure the energy balance was fulfilled the UA-values for each heat exchanger was estimated using the shell-and-tube heat exchanger model in Equation (3.7) on page 6 along with $Q = m_c c_{p,c}(T_{c,out} - T_{c,in}) = m_h c_{p,h}(T_{h,in} - T_{h,out})$. Two UA-values was then found, one using the cold energy balance and one using the hot energy balance, and the average was used. Both the new UA-values and the UA-values received from Statoil Mongstad can be found in Table F.1 in Appendix F. From the table it can be seen that the deviations in the UA-values are negligible.

7.2 Self-Optimizing Variables

The self-optimizing variables are

$$c_A = f_A - f_F \tag{7.1}$$
$$c_B = f_B - f_F \tag{7.2}$$
$$c_C = f_C - f_F \tag{7.3}$$
$$c_D = f_D - f_F \tag{7.4}$$
$$c_E = f_E - f_F \tag{7.5}$$
$$\tag{7.6}$$

where

$$f_A = \frac{T_{1A}^2}{T_{h,1A}^{in}} \tag{7.7}$$

$$f_B = \left(\frac{T_{h,2B}^{in} - T_{2B}}{T_{h,1B}^{in}} - 1\right)\frac{T_{1B}^2}{T_{h,2B}^{in} - T_{1B}} + \frac{T_{2B}^2}{T_{h,2B}^{in} - T_{1B}} \tag{7.8}$$

$$f_C = \left(\frac{T_{h,2C}^{in} - T_{2C}}{T_{h,1C}^{in}} - 1\right)\frac{T_{1C}^2}{T_{h,2C}^{in} - T_{1C}} + \frac{T_{2C}^2}{T_{h,2C}^{in} - T_{1C}} \tag{7.9}$$

$$f_D = \left(\frac{T_{h,2D}^{in} - T_{2D}}{T_{h,1D}^{in}} - 1\right)\frac{T_{1D}^2}{T_{h,2D}^{in} - T_{1D}} + \frac{T_{2D}^2}{T_{h,2D}^{in} - T_{1D}} \tag{7.10}$$

$$f_E = \frac{T_{1E}^2}{T_{h,1E}^{in}} \tag{7.11}$$

$$f_F = \left(\frac{T_{h,2F}^{in} - T_{2F}}{T_{h,1F}^{in}} - 1\right)\frac{T_{1F}^2}{T_{h,2F}^{in} - T_{1F}} + \frac{T_{2F}^2}{T_{h,2F}^{in} - T_{1F}} \tag{7.12}$$

Also here all the variables should be subtracted the inlet temperature, T_0, which is measured to be 132.91°C. The five controlled variables are implemented as five equality constraints in the function *fmincon* in `MATLAB`. In addition the constraints $m_G = 103.5$ tonnes/h and sum$(u) = 1$ (where u is a vector containing all the seven mass fractions) are implemented to assure that the flow through stream G is kept constant and the mass balance is fulfilled, respectively. The *fmincon* function varies the splits (or, equivalent, the mass flows) until all the constraints are equal to zero (or as close as possible). Running the m-file `mongstadSolve.m` (see Appendix G.3) the self-optimizing splits are calculated along with the new outlet temperaures of each cold stream.

7.3 Results

The self-optimizing splits, together with the RTO splits calculated from received data, are presented in Table 7.2. The RTO and SOC temperatures are presented in Table 7.3. The optimizer reports that four iterations and thirty-five function evaulations were required to find the optimum. The value of the maximum constraint (c_C) is reported to be -1.2×10^{-8} which indicates that all the constraints for any practical reasons are zero.

The temperature T in Table 7.3 is the temperature *before* heat exchanger O1 and T_{end} is the temperature *after* mixing with stream G, in accordance with Figure 7.2.

	RTO	SOC	Difference	Difference [%]
u_A	0.0944	0.0990	+0.0046	+4.8
u_B	0.1902	0.1989	+0.0088	+4.6
u_C	0.0699	0.0822	+0.0122	+17.5
u_D	0.1396	0.1469	+0.0072	+5.2
u_E	0.1360	0.1281	-0.0080	-5.8
u_F	0.2571	0.2323	-0.0248	-9.7
u_G	0.1127	0.1127	0	0

Table 7.2: RTO and self-optimizing splits

	RTO [°C]	SOC [°C]	Difference [°C]
T_{1A}	234.0	230.5	-3.5
T_{2B}	214.0	211.5	-2.5
T_{2C}	215.7	206.2	-9.5
T_{2D}	210.9	208.6	-2.3
T_{E1}	213.4	216.7	+3.3
T_{2F}	204.7	209.4	+4.7
T_{2G}	234.5	234.5	0
T	213.0	212.9	-0.1
T_{end}	224.7	224.6	-0.1

Table 7.3: Cold outlet temperatures

Comparing the RTO and SOC case, the table shows that there is a decrease in temperature out on stream A-D while there is an increase in temperature on stream E and F. The large decrease in temperature on stream C is explained by the large increase in mass fraction (+17.5 %).

According to this study using the proposed SOC strategy decreases T_{end} by 0.1°C. Since 0.1°C is a very small number the SOC and RTO performance are in practice the same. Also, since temperatures can not be measured exact one can not know precisely what the result is unless the self-optimizing control strategy is implemented on the plant. It can also be concluded that the optimum is flat since the SOC splits are a bit different from the RTO splits, but the the end temperatures are almost the

same.

The Mongstad HEN is a complicated network with some interconnection between the different hot streams. For instance, the self-optimizing control strategy suggests to increase u_D by 5.2 % which results in a lower hot temperature out, $T_{h,2D}$. According to the Mongstad HEN shown in Appendix E this stream (BSR12) is connected with the hot stream in on G2 (BSR13). The temperature on this stream will then also be higher which would justify an increase in the flow through stream G. Also, the hot inlet stream on F2 and F3 is used to further heat the crude oil in O1 after the streams A-F have been mixed. If the temperature of the crude oil is lower the hot inlet temperature on F2 and F3 will also be lower and would hence justify the decrease u_F. However, since the crude oil is only about 0.1°C lower this effect will be negligible small.

In between $T_{h,C2}$ and $T^{in}_{h,C1}$ (HGO2 and HGO3, see Appendix E) there is a drier which cools the liquid with about 1°C. The ΔT over the drier is assumed to be constant.

7.4 Parallel Heat Exchangers

After the six splits have been optimized the split related to the two parallel heat exchangers F2 and F3 could be optimized. However, these are not optimized in the RTO case (no controllers indicated, see Appendix E) and the streams are driven by the pressur in the pipes. Because of equal heat capacities for the hot streams and similar mass flows and UA-values for the two heat exchangers (110.7 kW/K and 112.2 kW/K) the split is close to 0.5, as indicated in Figure 7.3.

Finally, the split related to the H stream could also be optimized, but this split is neither optimized in the RTO case. If the split was to be optimized, it should be noted that the hot inlet of H2 is the hot outlet of H1. Thus, the controlled variable would be

$$c_H = \frac{T_1^2}{T_{h,1}^{in}} - \frac{T_2^2}{T_{h,1}}, \tag{7.13}$$

where subscript 1 refers to H1 and subscript 2 refers to H2.

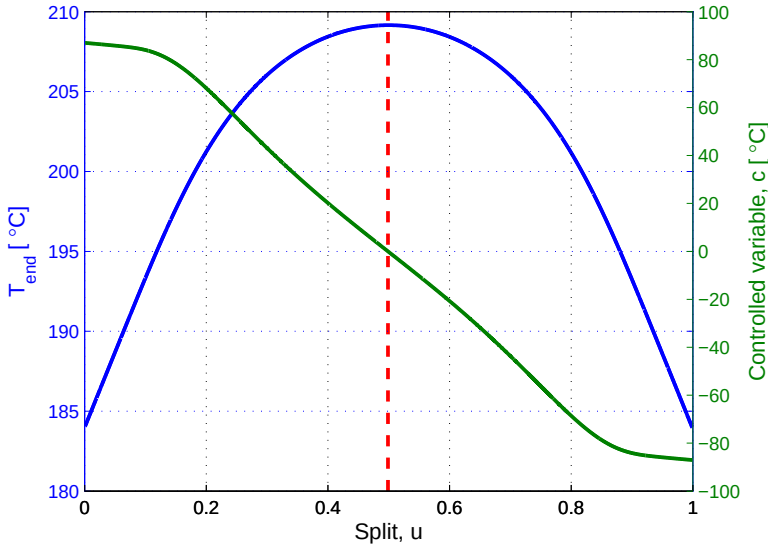


Figure 7.3: Optimal split between F2 and F3

It should be noted that there is a controller (`TIC319` in Appendix E) that controls the temperature after mixing with stream G, T_{end}, to 224.75°C. This means that if the temperature is lower than this more hot fluid will go through heat exchanger O1 (E153 in Appendix E) and hence less fluid will go through H1 and H2 (E174C/D and E174A/B in Appendix E, respectively), resulting in a lower end temperature. The argumentation is opposite for the case where the temperature after mixing with stream G is higher than 224.75°C. Hence, maximizing the temperature T_{end} is equivalent to maximizing the temperature after H1 and H2, which is really the temperature that should be maximized.

The SOC results are summarized in Figure 7.4.

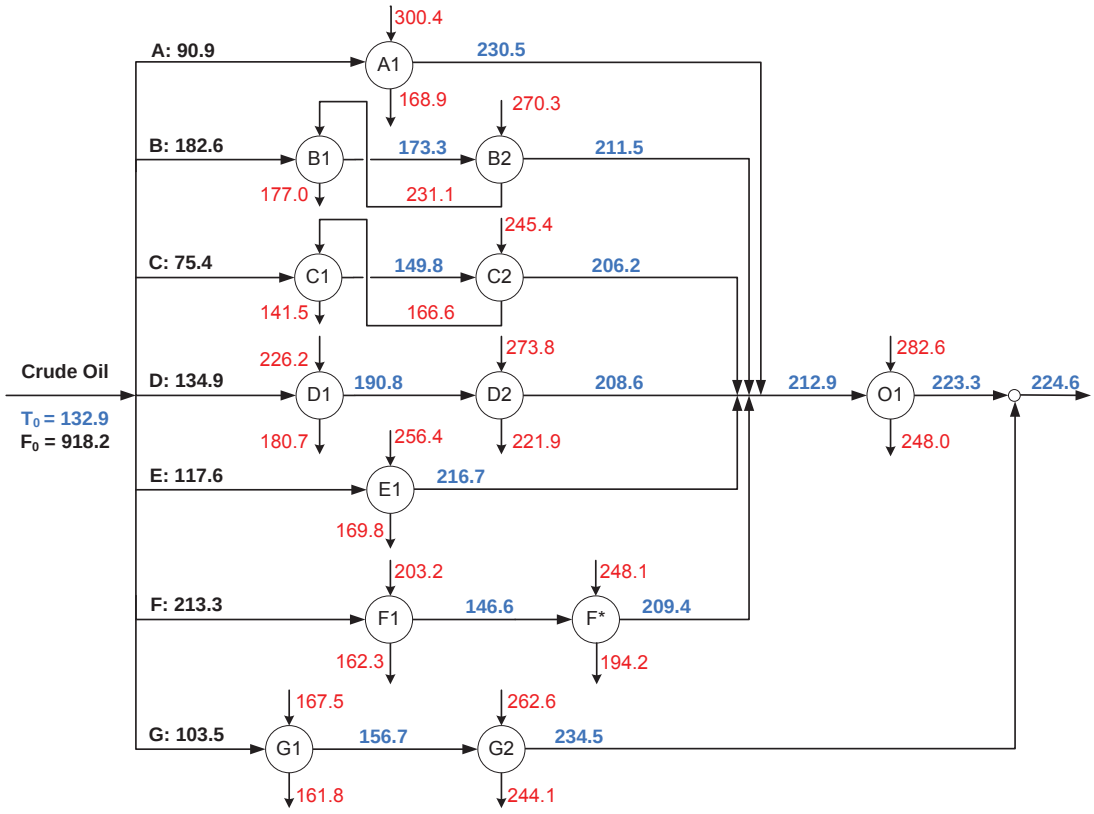


Figure 7.4: SOC Results

8 Self-Optimizing Control for LNG Plants

Optimal design of LNG plants has been extensively studied, but not optimal operation. Therefore it has been looked into if the proposed self-optimizing control strategy could be utilized in this wide technology area as well. Since the process is rather complicated it is believed that the simple controlled variable will not be sufficient. If it fails other methods could be used, for instance the exact local method. This optimization method will be described at the end of the chapter.

Only a brief introduction to the LNG technology will be given here. There are many papers disussing this; for an overview, see Edvardsen (2010).

8.1 Introduction

Liquefied Natural Gas (LNG) is, as the name indicates, cooled and condensed natural gas at approximate atmospheric pressure and about -160°C. Depending on the natural gas feed the mixture consists of mainly methane (85-95 mole%) and heavier hydrocarbons. The mixture often contains some nitrogen as well.

When the distance from the gas field to existing piping infrastructure is too large, building a complete new pipeline is often not economical feasible. LNG is produced to make the transportation of natural gas more convenient; liquefying the gas reduces its volume by a factor of approximately 600 and the product can be shipped by LNG carriers or transport trucks. Another benefit is the greater flexibility to choose customers when the natural gas is not transported in a pipeline. The initial investment costs are high for LNG, but the transport costs are lower compared to pipelines (Pettersen, 2010).

To liquefy natural gas there are many different production principles, ranging from the simple PRICO process to complicated cascade processes using mixed refrigerants like Statoil-Linde's mixed-fluid cascade (MFC) technology. The liquefaction of natural gas can be divided in three stages: precooling, liquefaction and subcooling. Figure 8.1 shows the three stages where the red arrows indicate heat transfer.

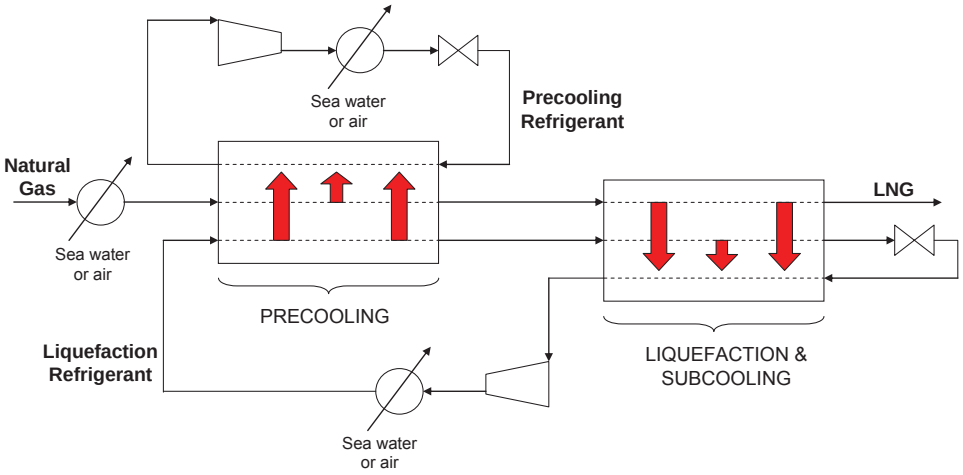


Figure 8.1: Simplified LNG process

Assuming inlet conditions for the natural gas of 60 bara and 20°C, the gas is precooled to about -50°C, condensed, liquefied and subcooled to about -150°C. During this cooling process there is only a small pressure drop related to transport in pipes and heat exchangers, so the pressure must be let down to about atmospheric pressure after subcooling. This is done in the end flash process with a turbine and a Joule-Thomson valve. The turbine lets down the pressure to almost the boiling point to ensure that there is not a two-phase flow within the turbine. The end flash is important to remove nitrogen from the natural gas and it is often used as fuel gas. (Pettersen, 2010).

The C3MR process uses pure propane as precooling refrigerant and a mixed refrgierant in the liquefaction part consisting of mainly methane and ethane, but also some propane and nitrogen. C3MR is the most used LNG technology in the world and therefore this technology will be studied here.

8.2 C3MR Technology

In Figure 8.2 a simplified flow scheme of the C3MR process is shown. The precooling part consists of three (can also be two or four) heat exchangers in series; E1A, E1B and E1C. The precooling medium is superheated in the heat exchangers to ensure no liquid is introduced to the compressors. After compression to the saturation pressure in C1 the precooling medium is condensed by heat exchanging with sea

water or ambient air in CW1. The temperature is decreased by letting down the pressure in an expansion valve, the two-phase flow is separated and the liquid is introduced to the heat exchanger. Both the natural gas and the mixed refrigerant for the liquefaction part are precooled. After precooling the mixed refrigerant is a two-phase flow. The MR is separated in D1 and the heavier liquid fraction is subcooled in E2A and used to liquefy the natural gas, while the lighter vapour fraction is condensed and used to subcool the natural gas. The two heat exchangers E2A and E2B shown in the scheme are actually one spiral wound heat exchanger. (Pettersen, 2010).

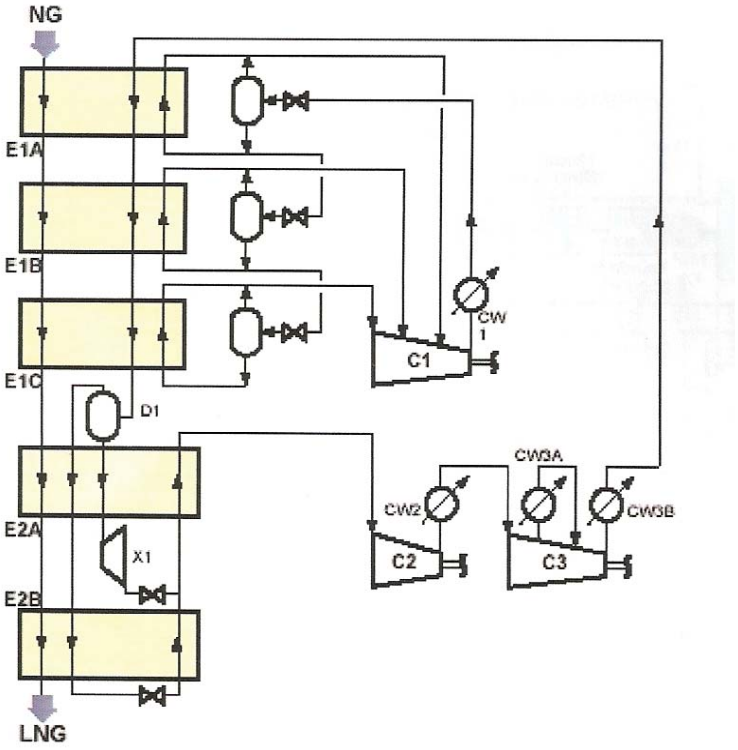


Figure 8.2: C3MR process

8.3 Self-Optimizing Control

Up to now the objective function to be minimized has been $J = -T_{end}$, but in this case the objective function is the opposite, $J = +T_{end}$, i.e. the end temperature should be minimized. In either case the optimization problem is the same; as long as the *heat transfer* is maximized the objective function is minimized.

It is only in the precooling part that stream splits are involved. In Figure 8.2 there are two stream splits; downstream the two first (upper) separators the liquid propane stream is split. This split could potentially be optimized using the self-optimizing control strategy proposed in this study. A simplified flowsheet for the precooling part is shown in Figure 8.3.

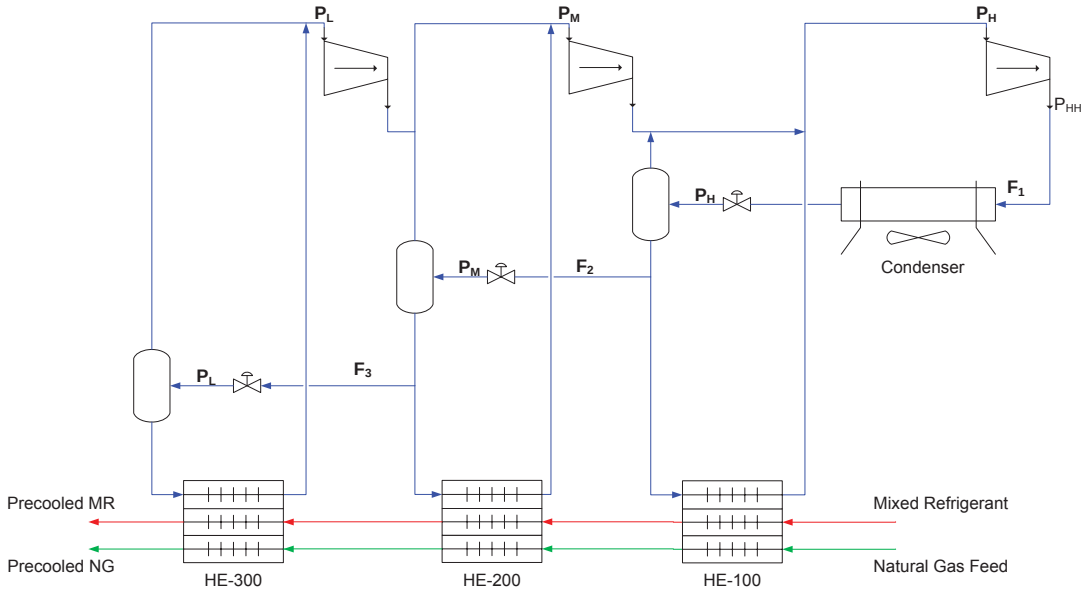


Figure 8.3: Precooling part of C3MR process

he objective function to be minimized is

$$\begin{aligned} \underset{u}{\text{minmize}} \quad & J = -W_s \\ \text{subject to} \quad & c \leq 0 \end{aligned} \tag{8.1}$$

where W_s is the compressor work and c are the constraints. The constraints are that there should be superheated propane vapour out of each heat exchanger to make sure that no liquid is introduced to the compressors.

The differences are many from the previous cases studied and the LNG/C3MR case. Some of the differences are:

- The cold fluid (propane) is evaporating in the heat exchangers (superheated)
- The natural gas and mixed refrigerant are partially condensing in the heat exchangers

- The cold inlet temperatures (propane) are different for each heat exchanger
- The pressures are different in each of the cold (propane) inlet streams

Thus, the model equations are not the same, and the proposed self-optimizing control strategy can not be used. A new invariant self-optimizing control variable would have to be found by putting up new energy balances. For one heat exchanger the energy balances are:

$$Q = m\Delta h_{vap} \tag{8.2}$$

$$Q = UA\Delta T_{LM} \tag{8.3}$$

where Δh_{vap} is the heat of vaporization and 0°C of superheating (saturated propane vapour) is assumed out of the heat exchanger. In practice, 5°C superheating is typically used. It is also assumed that neither the mixed refrigerant nor the natural gas is condensing in the heat exchanger. As a start, these might be natural assumptions to make. As before, ΔT_{LM} could be approximated with ΔT_{AM}. Since the inlet and outlet temperature of the cold propane stream are the same, the AMTD is

$$\Delta T_{AM} = \frac{T_{h,in} + T_{h,out} - 2T_{c,in}}{2} \tag{8.4}$$

However, it is unlikely that an approach using sparse resultants to eliminate variables would result in a convenient self-optimizing variable. Instead, one of the other methods mentioned in Chapter 3.2 could be used to achieve self-optimizing control, for instance the exact local method.

8.3.1 Exact Local Method

The exact local method was derived by Halvorsen et al. (2003). If W_d (diagonal matrix) contains the magnitude of disturbances and W_e contains the implementation errors assosciated with the controlled variables, then, assuming $||f||_2 = 1$ where $f = [d\ e]^T$ (d is the disturbance and e is the implementation error), the worst-case loss is

$$\max_{||f||_2 \leq 1} \quad L = \frac{1}{2}\bar{\sigma}([M_d \;\; M_e])^2 \tag{8.5}$$

where

$$M_d = J_{uu}^{1/2}(J_{uu}^{-1}J_{ud} - G^{-1}G_d)W_d \tag{8.6}$$

$$M_e = J_{uu}^{1/2}G^{-1}W_e \tag{8.7}$$

and $\bar{\sigma}$ is the upper singular value, J is the cost function, G and G_d are the steady-state gain matrix and disturbance model, respectively, and u degrees of freedom. In our case the degrees of freedom are the three flows F_1, F_2, F_3 and the three pressures P_H, P_M and P_L. P_{HH} is set since this is the pressure propane condensates at the given ambient temperature. G, G_d and W_e are dependent on the measurement combination matrix H through:

$$G = HG^y; \quad G_d = HG_d^y; \quad W_e = HW_e \tag{8.8}$$

Halvorsen et al. (2003) further propose a procedure to find the candidate set of controlled variables c:

1. Specify the cost function J ($J = W_s$ = Compressor work)
2. Solve the nominal optimization problem and find J_{uu} and J_{ud}
3. For each c, find the linear model $\Delta c = G\Delta u + G_d\Delta d$
4. Find W_d and W_e
5. For each c, compute the singular value $\bar{\sigma}([M_d \;\; M_e])$
6. The set c with the smalles singular value minimizes the loss in Equation (8.5)

The important disturbances in this case are the natural gas feed composition and flow rate, and feed temperature and pressure (both natural gas and mixed refrigerant). Here the mixed refrigerant is considered as a "feed" to the precooling section.

Solving the nominal optimization problem (point 2 in the procedure above) can be a difficult task for an LNG process. The process is extremely sensitive to changes in the manipulated variables (u's) and convergence failure is often a problem. However, if the optimization problem could be solved this method is a good approach to finding self-optimizing variables. It should be noted that the exact local method, as the name implies, is only valid around the point which is linearized.

9 Discussion and Further Work

This chapter is organized in three parts: A general part and two parts discussing Perstorp and Statoil Mongstad, respectively.

9.1 General

In this work ideal mixing was assumed to calculate the end temperatures of the different heat exchanger networks. When the streams have different temperature, an exergy loss is associated with the mixing. If the temperature difference between the streams increase the specific exergy of the mixture will remain unchanged, but the absolute losses will increase (Semenyuk, 1976). A controlled variable which would eliminate the exergy loss is $c = T_1 - T_2$ for a HEN with two heat exchangers in parallel. However, the controlled variable was investigated through simulations and it is revealed that the performance is not satisfactory, hence it is not a good candidate as self-optimizing variable.

Also, the reduced gradient of the objective function, $J_{z,red}$, has been used in the simulations as controlled variable. This results in optimal operation, but the reduced gradient includes measurements like UA, u, ω_0 and ω_h which we would avoid to measure.

9.2 Perstorp Study

It is obvious that some of the measured data from Perstorp are inconsistent. This is easily realized by investigating the temperature difference between the outlet temperature of the hot streams $T_{h,11}$, $T_{h,2}$ and $T_{h,3}$ and the cold inlet temperature T_0. This temperature difference should be positive since negative heat transfer is not possible. Similary, for HX1 the hot outlet temperature should be higher than the cold inlet temperature. However, for some of the data points this is not the case, and investigating the measurements for 2010 reveals that 557 of 8760 measurements (6.8%) show this behaviour.

The heat exchangers in the Perstorp HEN are three spiral plate heat exchangers and one plate and fin heat exchanger. In this study the flow pattern is assumed to

be countercurrent which is approximately correct for a spiral plate heat exchanger. However, the heat transafer is not pure counter-flow (Picon-Nunez et al., 2007). In intermediate turns the hot stream exchanges heat with two adjacent cold streams at different temperature. Hence, the temperature driving forces are lower than ΔT_{LM}. To improve the Perstorp model a correction factor F_S could be introduced in the heat exchagner model $Q = UA\Delta TF_S$, where F_S was determined by Bes and Roetzel (1993):

$$F_S = \frac{\ln(1+CN^2}{CN^2}) \tag{9.1}$$

$$CN = 2\sqrt{\frac{A}{\pi A_c} NTU_c NTU_h} \tag{9.2}$$

where CN is referred to as the criterion number. If the hot and cold heat transfer area are the same the criterion number reduces to

$$CN = \frac{2}{\sqrt{\pi}} NTU \tag{9.3}$$

If NTU is replaced by $NTU \times F_S$ the efficiency is defined as in Equation (3.22). The hot and cold outlet temperatures can then be found by using Equation (3.26) and Equation (3.27), respectively.

The plate and fin heat exchanger model should also be improved since cross flow is the flow pattern here. The equations that could be used are referred to in many papers, for instance Sanaye and Hajabdollah (2010). This would ensure that a more accurate model is used resulting in a more describing end temperature. However, from the steady-state analysis that was conducted it was discovered that the self-optimizing variable performed well in all the cases. Thus, it is likely to believe that the performance is equally good if a more accurate model is used. It was stated that the potential for improvement is about 1-3°C. Since the optimal temperature will be lower, because of the lower driving force $\Delta T_{LM} F_S$, the potential for improvement will also be lower. To quantify this value new simulations should be performed. This is suggested as further work.

9.3 Statoil Mongstad Study

It is assumed that the RTO performance is close to optimum, but the real optimum is not known. However, this could be found by using *fmincon* in `MATLAB`. The objective function should be implemented as $J = -T_{end}$ and the constraints should be removed, besides of the $sum(u) = 1$ constraint.

A dynamic model was not made for the Mongstad study. This is left as further work.

The Mongstad model could also be improved further by performing some more detailed C_p-calculations. Since the cold and hot fluids are all hydrocarbons the heat capacity's dependency on temperature is easily determined. In this study the C_p has been calculated from received data from Statoil Mongstad and assumed constant throughout the heat exchanger. C_p was calculated by taking the average of $C_{p,in}$ and $C_{p,out}$ on each heat exchanger, both for cold and hot side. A better approach would be to let the C_p vary with the corresponding mass flows (and hence temperatures) for each heat exchanger. The `MATLAB` function *fsolve* could be used for this purpose. By guessing an outlet $C_{p,0}$ value the average could be used to calculate the outlet temperatures and then calculating again the outlet C_p. *fsolve* then varies the *guessed* outlet C_p value until it matches with the *calculated* C_p value. This is left as further work.

9.3.1 Improved approximation

Recall that for a countercurrent heat exchanger, if $1/1.4 < \Theta_1/\Theta_2 < 1.4$ the error is less than 1% if LMTD is replaced by AMTD. When shell-and-tube heat exchangers are used the correction factor F must be included and the relevant approximation is $\Delta T_{LM} F \approx \Delta T_{AM}$. Since $\Delta T_{LM} < \Delta T_{AM}$ and $F \leq 1$ this approximation will differ more than the approximation $\Delta T_{LM} \approx \Delta T_{AM}$.

By looking at the Equations (3.8)-(3.12) it is obvious that F is dependent on the hot and cold temperatures, not only Θ_1 and Θ_2. F is further dependent on R, P and N, which means that if all the four temperatures (hot and cold inlet and outlet) change by the same factor k, both R and P will remain unchanged and the same number of shells will be required to achieve the same correction factor. Both the

LMTD and the AMTD will increase by the factor k resulting in the same error as in the original case.

The higher temperature levels the higher is the approximation error for the same number of shells. As an example, if the hot and cold inlet and outlet temperatures are 20°C, 10°C, 7°C and 18°C, respectively, and the number of shells are 4, the error associated with the approximation $\Delta T_{LM} F \approx \Delta T_{AM}$ is 32 %. If the hot and cold inlet and outlet temperatures are increased to 200°C, 100°C, 97°C and 198°C, respectively, the number of shells must be increased to about 38 to end up with the same approximation error. This is clearly an unrealistic number of shells, but illustrates the temperature dependency. Note that both Θ_1 and Θ_2 are equal for the two cases.

It is obvious that the approximation can result in high errors in some cases. The errors can potentially counteract each other, but that is just a matter of being lucky or unlucky. If $k < 1$ is a constant and of similar size as F one could think that the approximation

$$F\Delta T_{LM} \approx k\Delta T_{AM} \tag{9.4}$$

would improve the performance since $F \leq 1$. Typically F is in the range $0.8 < F < 1$ which means that k could be chosen to by equal to 0.9. This will reduce the approximation error where the original approximation is not good, but it will increase the error where the original approximation is good. As an example, take the errors listed in Table 7.1 on page 49. The absolute error, $\sqrt{\sum Error_i^2}$, is 45.6 %, where i represents the individual stream. If $k = 0.9$ was used the absolute error would be reduced to 27.4 %. The minimum absolute error for a constant k is for $k = 0.915$, resulting in an absolute error of 26.6 %.

However, even if we model the heat transfer by $Q = UAk\Delta T_{AM}$ the self-optimizing variable stays the same. Recall from Chapter 4.1 that the UA values were eliminated even if it was not included in the list of variables to be eliminated. Hence, since UA was eliminated, also the product kUA would be eliminated. In other words, the size of k is not of importance in order to find a good self-optimizing variable.

9.3.2 Include stream G

When optimizing the crude unit heat exchanger network stream G was kept constant. The complete network could be optimized by deriving a new model. It could be imagined that the heat exchanger O1 was direcly installed on each stream A-F, where the corresponding UA-value would be

$$UA_i = UA_{\mathrm{O1}} u_i \qquad i = A, B \ldots F \tag{9.5}$$

where u is the mass fraction (not including stream G) on each stream. On stream B, C, D and F there would then be three heat exchangers in series and a new steady-state model (g in Equation (8.1) on page 59) must be found using the same approach as described in Chapter 3.4. This is left as further work.

10 Conclusion

In this study a self-optimizing variable has been applied on several cases. The performance has been satisfactory in all the cases. With the proposed self-optimizing strategy the performance at Perstorp can be improved according to the simulations. The performance at Statoil Mongstad is in practice the same using RTO and the proposed SOC strategy.

Advantages with the method is that it relies only on simple temperature measurements, hence no flow measurements are needed nor technical data like the heat exchanger area, heat transfer coefficients and heat capacities are necessary. The performance is best for well designed processes where ΔT_{AM} is close to ΔT_{LM}. The control structure is simple since only PI controllers are needed. With well tuned controllers good disturbance rejection can be achieved.

The self-optimizing variable is only used for relatively simple heat exchanger networks. For more compliated HENs, for instance the ones we find in LNG plants, further work is needed.

Bibliography

Aguilera, N., Marchetti, J. L., 1998. Optimizing and Controlling the Operation of Heat-Exchanger Networks. American Institute of Chemical Engineers Journal 44 (5), 1090 – 1104.

Bes, T., Roetzel, W., 1993. Thermal Theory for Spiral Heat Exchanger. International Journal of Heat and Mass Transfer 36 (3), 765–773.

Bjork, K.-M., Nordman, R., 2005. Solving large-scale retrofit heat exchanger network synthesis problems with mathematical optimization methods. Chemical Engineering and Processing 44 (8), 869 – 876.

Boyaci, C., Uzturk, D., Konukman, A. E. S., Akman, U., 1996. Dynamics and Optimal Control of Flexible Heat-Exchanger Networks. Computers and Chemical Engineering 20 (Supplement 2), 775 – 780.

Buse, L., Mourrain, B., June 1998. multires. Version last modified 08/2002. URL `http://www-sop.inria.fr/galaad/software/multires/`

Chachuat, B., Srinivasan, B., Bonvin, D., 2009. Adaptation Strategies for Real-Time Optimization. Computers and Chemical Engineering 33 (10), 1557 – 1567.

Chen, J. J. J., 1987. Comments on improvements on a replacement for the logarithmic mean. Chemical Engineering Science 42 (10), 2488–2489.

Edvardsen, D. G., 2010. Simulation, Design and Optimal Operation of LNG Process for Arctic Conditions. Specialization Project. NTNU, Trondheim.

Glemmestad, B., Skogestad, S., Gundersen, T., 1999. Optimal operation of heat exchanger networks. Computers and Chemical Engineering 23 (4-5), 509 – 522.

Gorji-Bandpy, M., Yahyazadeh-Jelodar, H., Khalili, M., 2011. Optimization of heat exchanger network. Applied Thermal Engineering 31 (5), 779 – 784.

Halvorsen, I. J., Skogestad, S., 1997. Indirect on-line optimization through setpoint control. Prepared for presentation at the AIChe 1997 Annual Meeting .

Halvorsen, I. J., Skogestad, S., Morud, J. C., Alstad, V., 2003. Optimal Selection of Controlled Variables. Industrial & Engineering Chemistry Research 42, 3273–3284.

Incropera, F. P., Dewitt, D. P., Bergman, T. L., Lavine, A. S., 2007. Introduction to Heat Transfer, 2nd Edition. John Wiley & Sons, New York.

Jäschke, J., 2011. Invariants for Optimal Operation of Process Systems. Dissertation, Norwegian University of Science and Technology.

Jäschke, J., Skogestad, S., 2011a. Self-Optimizing Control for Polynomial Systems. Preprint submitted to Journal of Process Control .

Jäschke, J., Skogestad, S., 2011b. Simple Methods to Maximise Heat Transfer in Heat Exchanger Networks - Internal Report .

Lid, T., Skogestad, S., 2001. Implementation issues for real-time optimization of a crude unit heat exchanger network. In: Gani, R., Jorgensen, S. B. (Eds.), European Symposium on Computer Aided Process Engineering - 11, 34th European Symposium of the Working Party on Computer Aided Process Engineering. Vol. 9 of Computer Aided Chemical Engineering. Elsevier, pp. 1041 – 1046.

Lin, B., Miller, D. C., 2004. Solving heat exchanger network synthesis problems with Tabu Search. Computers and Chemical Engineering 28 (8), 1451 – 1464.

Linnhoff, B., Hindmarsh, E., 1983. The pinch design method for heat exchanger networks. Chemical Engineering Science 38 (5), 745 – 763.

Mathisen, K. W., Morari, M., Skogestad, S., 1994a. Dynamic models for heat exchangers and heat exchanger networks. Computers and Chemical Engineering 18 (1), 459–463.

Mathisen, K. W., Morari, M., Skogestad, S., Wolff, E. A., 1994b. Optimal Operation of Heat Exchanger Networks Presented at Process Systems Engineering (PSE'94), Kyongju, Korea.

Mathisen, K. W., Skogestad, S., Wolff, E. A., 1992. Bypass Selection for Control of Heat Exchanger Networks. Computers and Chemical Engineering 16 (Supplement 1), 263 – 272.

Novazzi, L. F., Zemp, R. J., 2009. Optimal Control of Heat Exchanger Networks. In: Rita Maria de Brito Alves, C. A. O. d. N., Evaristo Chalbaud Biscaia, J. (Eds.), 10th International Symposium on Process Systems Engineering: Part A. Vol. 27 of Computer Aided Chemical Engineering. Elsevier, pp. 1647 – 1652.

Paterson, W. R., 1984. A replacement for the logarithmic mean. Chemical Engineering Science 39 (11), 1635–1636.

Pettersen, J., August 2010. Compendium LNG Technology (TEP4185 Industrial Process and Energy Technology). Trondheim.

Picon-Nunez, M., Canizalez-Dvalos, L., Martnez-Rodrguez, G., Polley, G. T., 2007. Shortcut Design Approach for Spiral Heat Exchangers. Food and Bioproducts Processing 85 (4), 322 – 327.

Sanaye, S., Hajabdollah, H., 2010. Thermal-economic multi-objective optimization of plate fin heat exchanger using genetic algorithm. Applied Energy 87 (6), 1893–1902.

Seborg, D. E., Edgar, T. F., Mellichamp, D. A., 2003. Process Dynamics and Control, 2nd Edition. John Wiley & Sons, New York.

Semenyuk, L. G., 1976. Analysis of exergy loss in working-body mixing. Sanitary Engineering 16, 43–46.

Serth, R. W., 2007. Process Heat Transfer - Principles and Applications, 1st Edition. Elsevier Ltd., Oxford.

Sinnott, R., Towler, G., 2009. Chemical Engineering Design, 5th Edition. Elsevier Ltd., Oxford.

Skogestad, S., 2000. Plantwide control: the search for the self-optimizing control structure. Journal of Process Control 10.

Skogestad, S., 2003a. Prosessteknikk, 2nd Edition. Tapir Akademisk Forlag, Trondheim.

Skogestad, S., 2003b. Simple analytic rules for model reduction and PID controller tuning. Journal of Process Control 13, 291–309.

Skogestad, S., 2004. Near-optimal operation by self-optimizing control: from process control to marathon running and business systems. Computers and Chemical Engineering 29, 127–137.

Skogestad, S., Postlethwaite, I., 2005. Multivariable Feedback Control, 2nd Edition. John Wiley & Sons, Chichester.

Statoil, May 2011. The Mongstad Facility. `http://www.statoil.com/en/OurOperations/TerminalsRefining/ProdFacilitiesMongstad/Pages/default.aspx`.

Underwood, A. J. V., 1933. Graphical computation of logarithmic mean temperature difference. Industrial Chemist and Chemical Manufacturer 9, 167–170.

White, D. C., 1997. Online optimization: What, where and estimating ROI. Hydrocarbon Processing 76 (6), 4351.

A Case Studies (I)

A.1 Case II-a: $\omega_h < \omega_c$ and high UA-values

	ω_h [W/K]	$T_{h,in}$ [°C]	UA [W/K]
HX1	30	120	120
HX2	50	140	200

Table A.1: Parameters for Case II-a

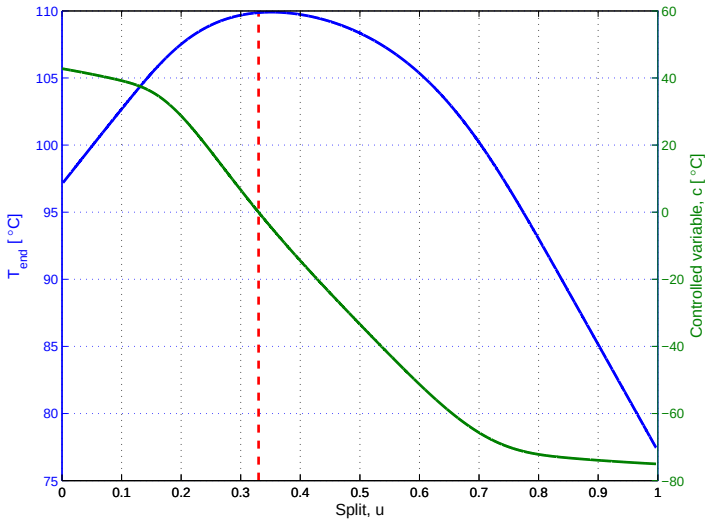
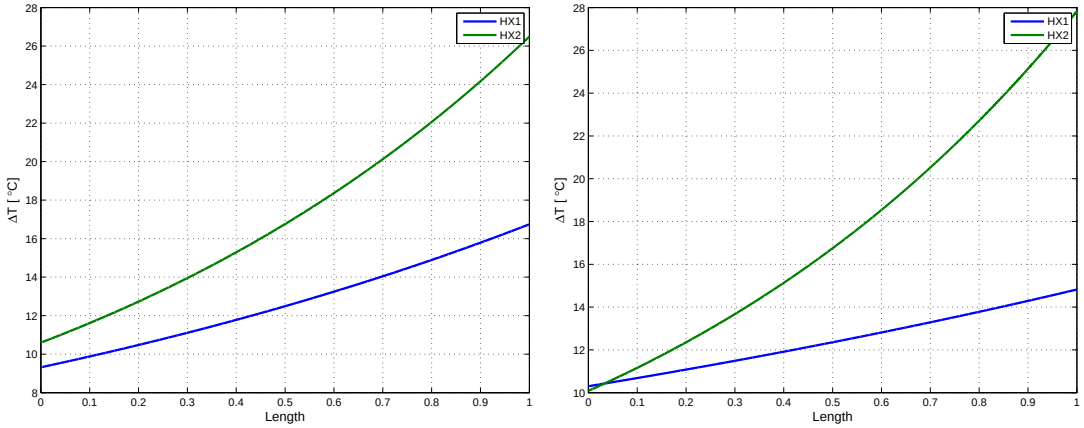
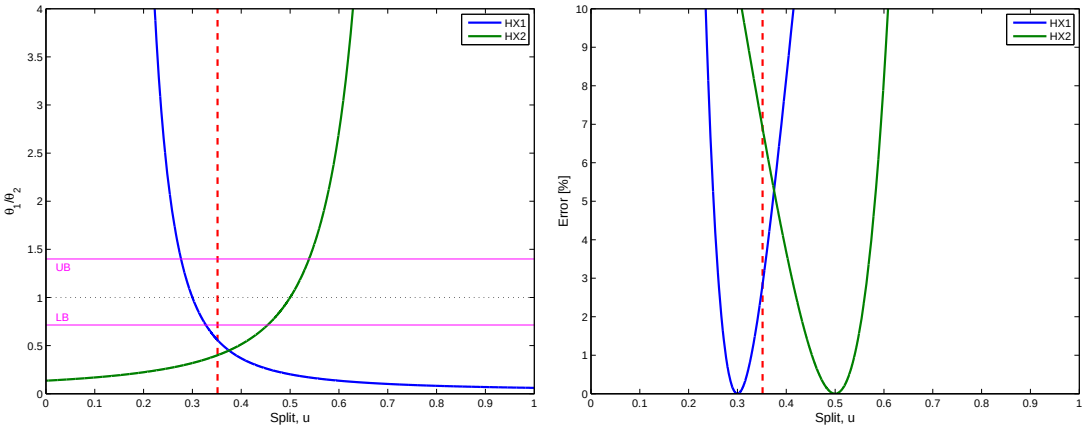


Figure A.1: T and c vs split



(a) Optimal (b) Self-optimizing

Figure A.2: Temperature profiles



(a) Approximation (b) Error

Figure A.3: Approximation

A.2 Case III-a: $\omega_h = \omega_c$ and low UA-values

	ω_h [W/K]	$T_{h,in}$ [°C]	UA [W/K]
HX1	50	120	50
HX2	50	140	80

Table A.2: Parameters for Case IV-a

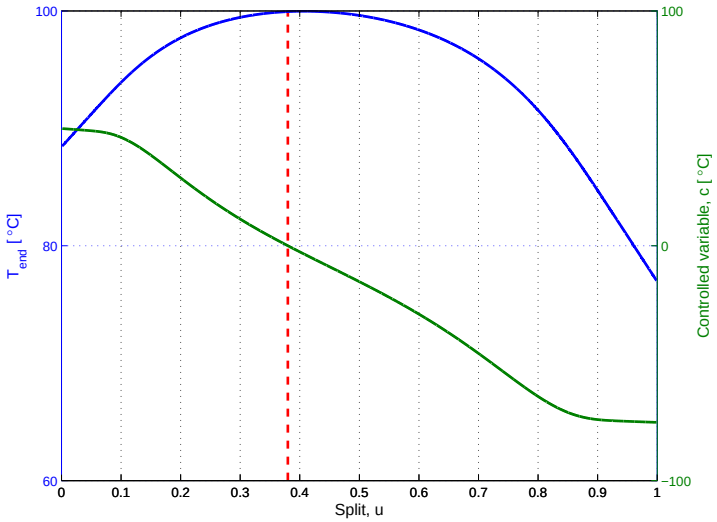
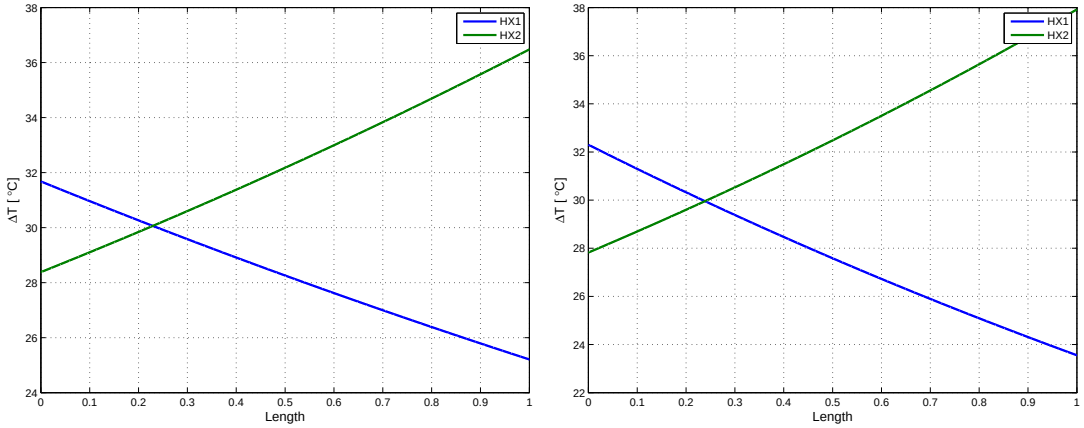
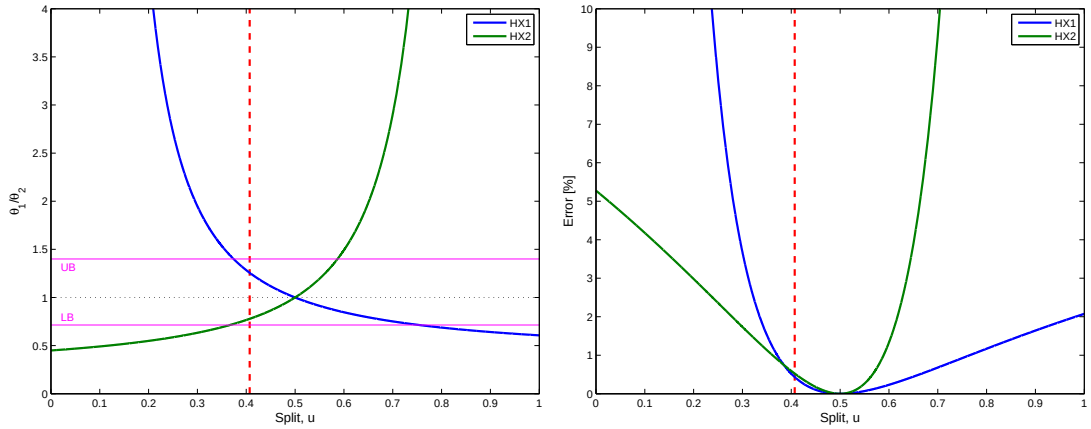


Figure A.4: T and c vs split



(a) Optimal (b) Self-optimizing

Figure A.5: Temperature profiles



(a) Approximation (b) Error

Figure A.6: Approximation

A.3 Case IV-a: $\omega_h = \omega_c$ and low UA-values

	ω_h [W/K]	$T_{h,in}$ [°C]	UA [W/K]
HX1	50	120	120
HX2	50	140	200

Table A.3: Parameters for Case V-a

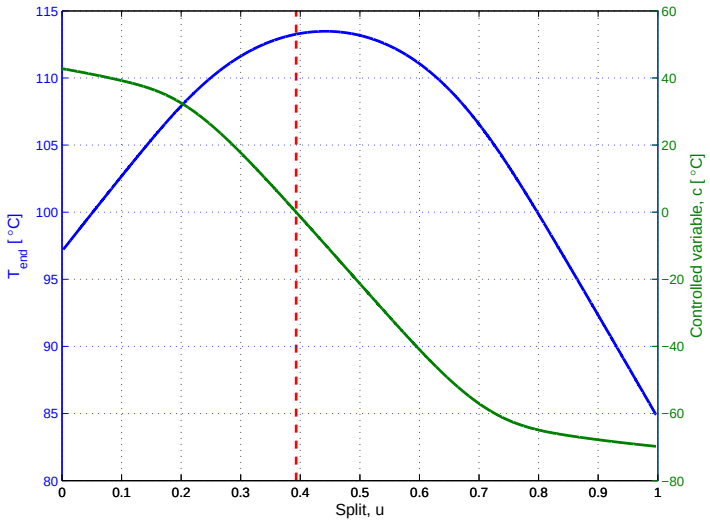
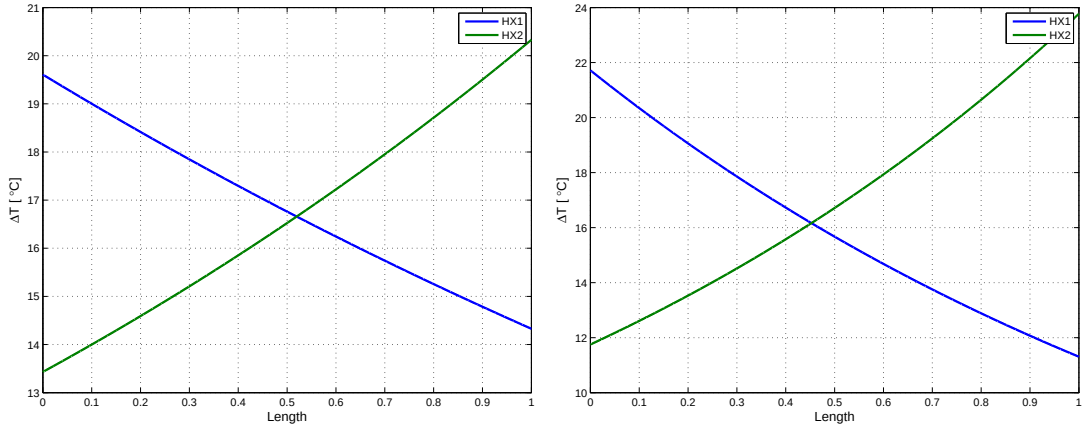
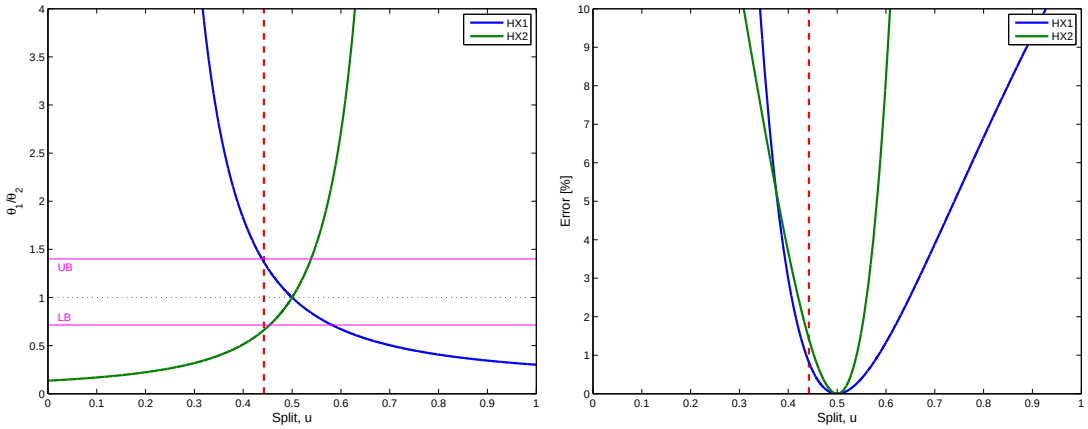


Figure A.7: T and c vs split



(a) Optimal (b) Self-optimizing

Figure A.8: Temperature profiles



(a) Approximation (b) Error

Figure A.9: Approximation

A.4 Case V-a: $\omega_h > \omega_c$ and low UA-values

	ω_h [W/K]	$T_{h,in}$ [°C]	UA [W/K]
HX1	80	120	120
HX2	60	140	200

Table A.4: Parameters for Case VI-a

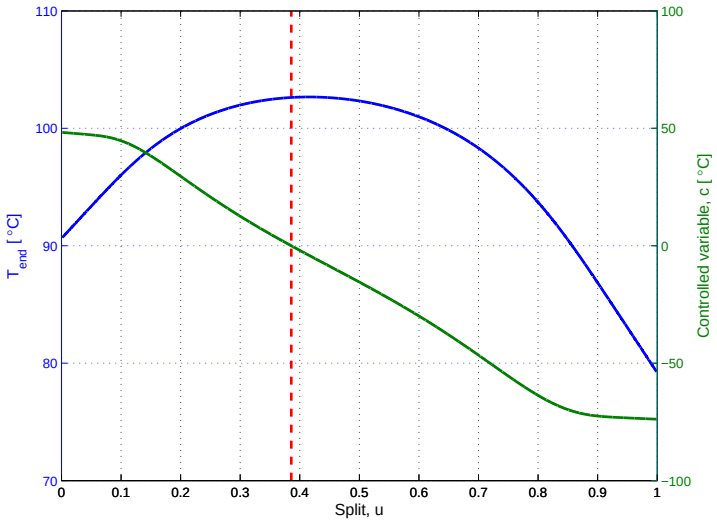
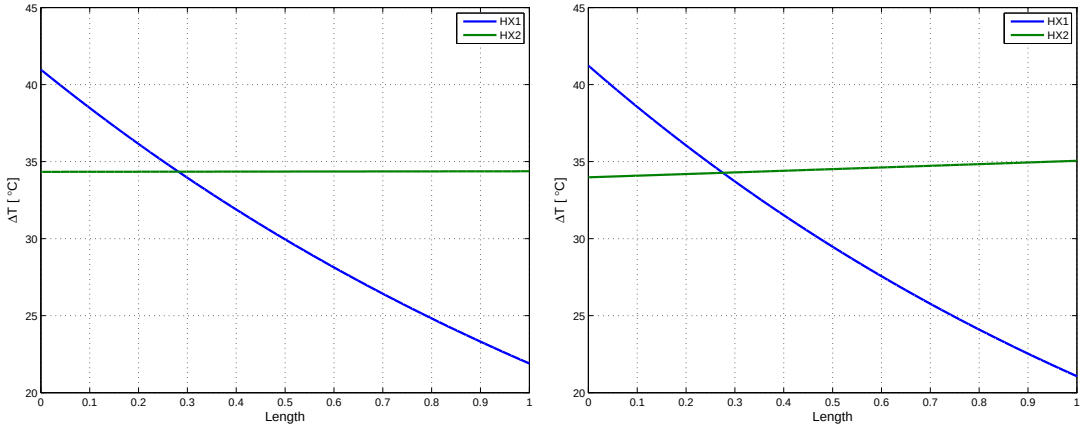
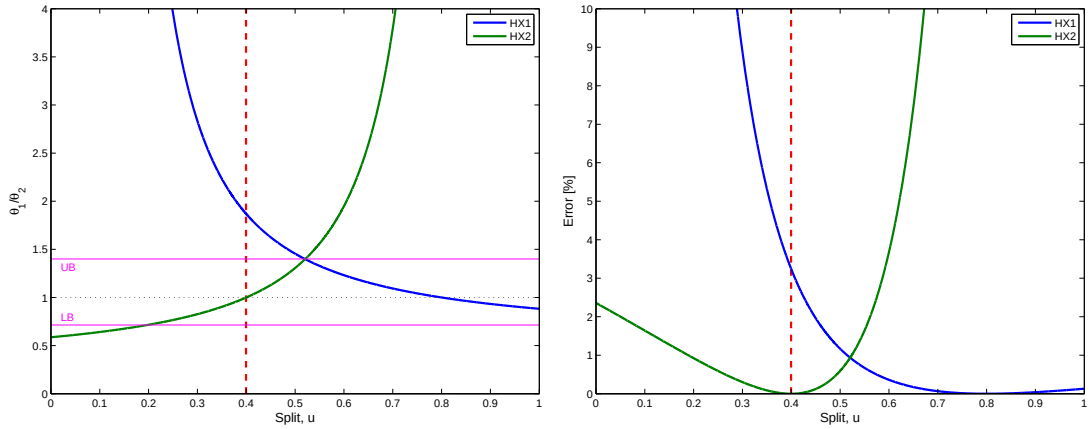


Figure A.10: T and c vs split



(a) Optimal (b) Self-optimizing

Figure A.11: Temperature profiles



(a) Approximation (b) Error

Figure A.12: Approximation

A.5 Case VI-a: $\omega_h > \omega_c$ and high UA-values

	ω_h [W/K]	$T_{h,in}$ [°C]	UA [W/K]
HX1	80	120	120
HX2	60	140	200

Table A.5: Parameters for Case VII-a

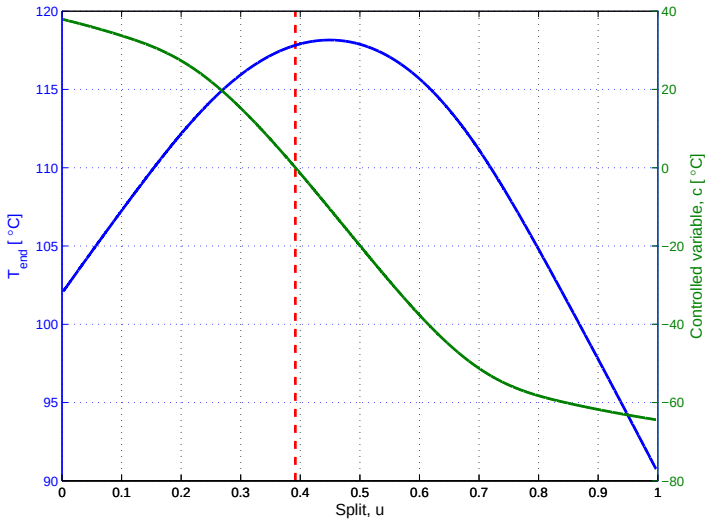
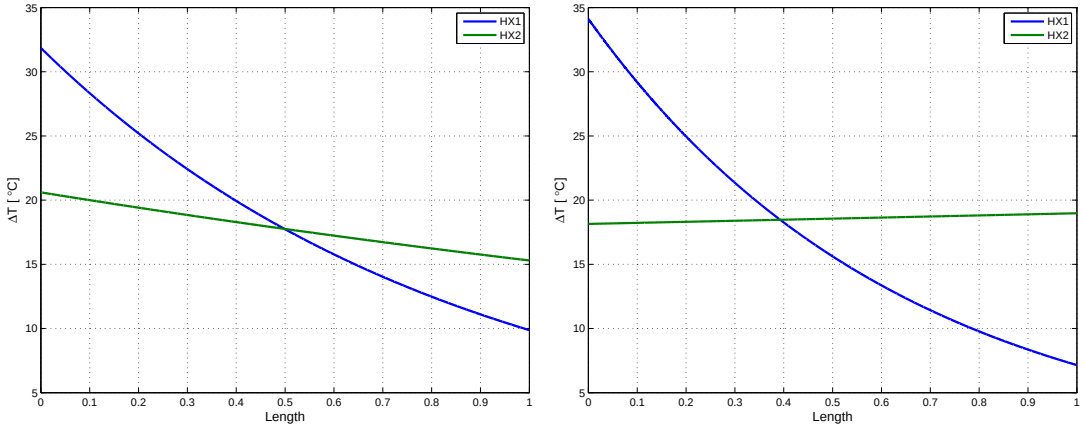
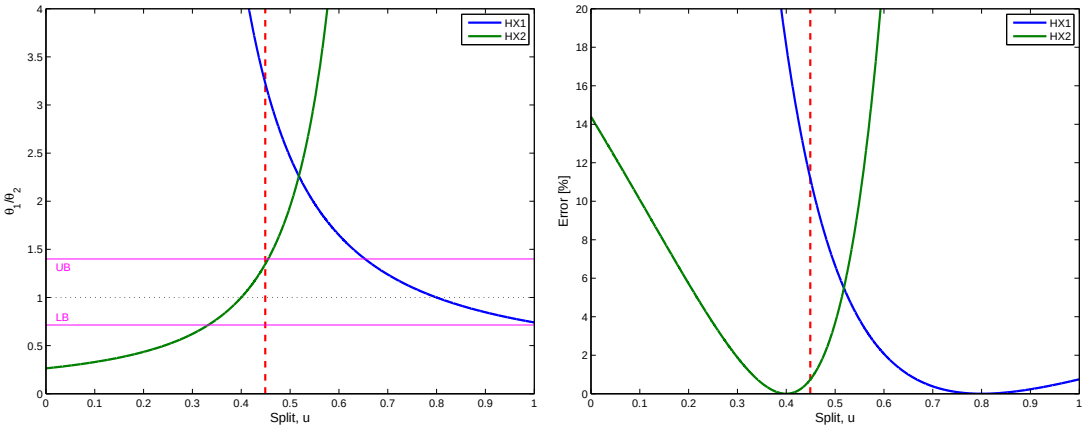


Figure A.13: T and c vs split



(a) Optimal (b) Self-optimizing

Figure A.14: Temperature profiles



(a) Approximation (b) Error

Figure A.15: Approximation

A.6 Case VII-a: $\omega_h > \omega_c$ and high UA-values

	ω_h [W/K]	$T_{h,in}$ [°C]	UA [W/K]
HX1	80	120	95
HX2	60	140	110

Table A.6: Parameters for Case VIII-a

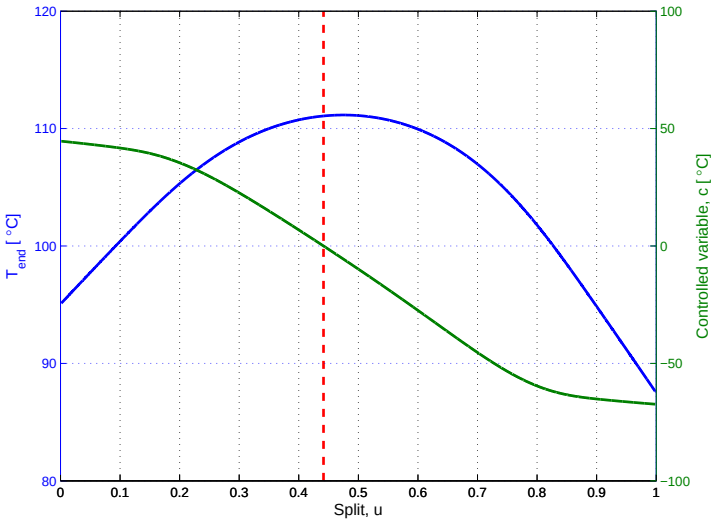
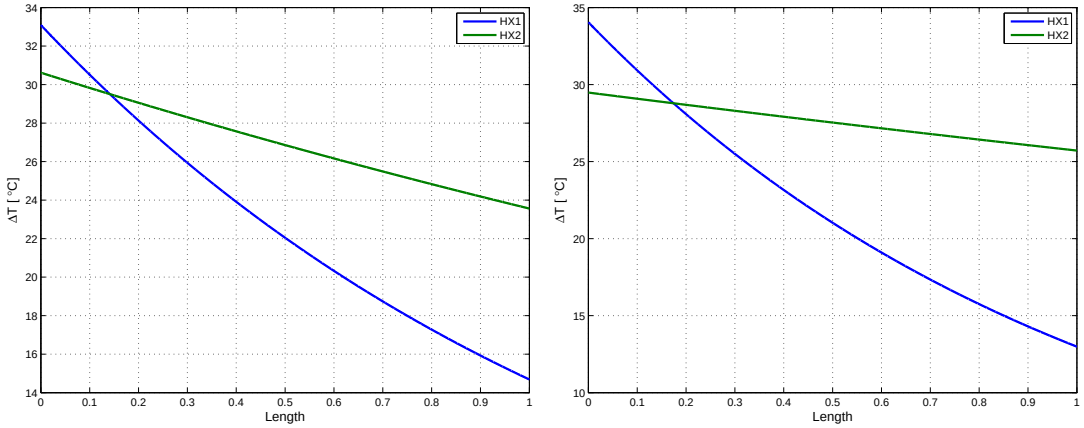
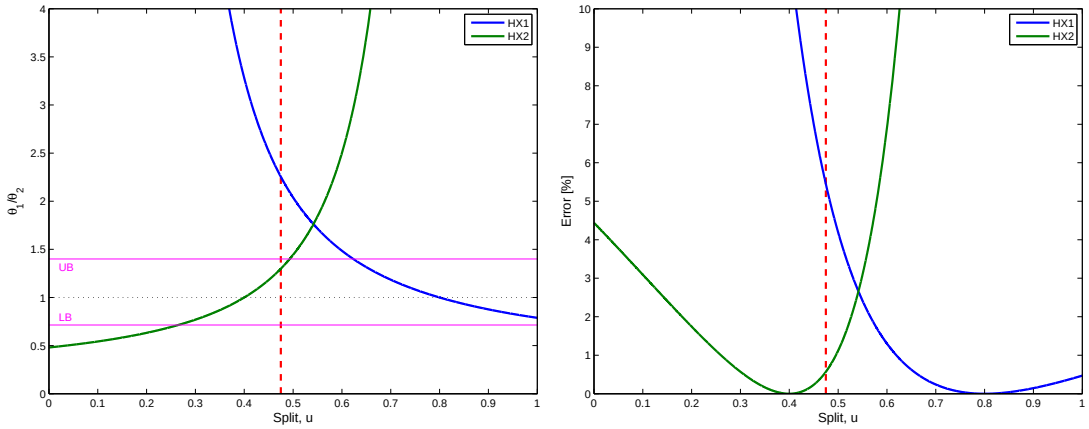


Figure A.16: T and c vs split



(a) Optimal (b) Self-optimizing

Figure A.17: Temperature profiles



(a) Approximation (b) Error

Figure A.18: Approximation

B Case Studies (II)

B.1 Case II-b: $\omega_h > \omega_c$ and high UA-values

	ω_h [W/K]	$T_{h,in}$ [°C]	UA [W/K]
HX11	30	120	100
HX1	50	140	180
HX2	40	140	60

Table B.1: Parameters for Case II-b

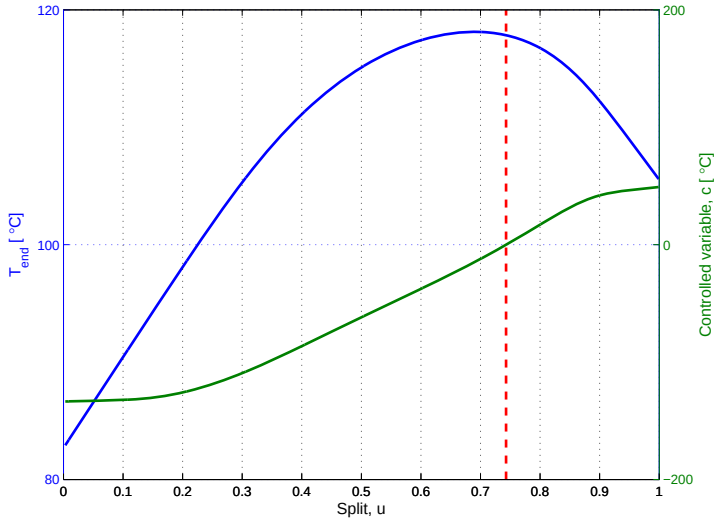
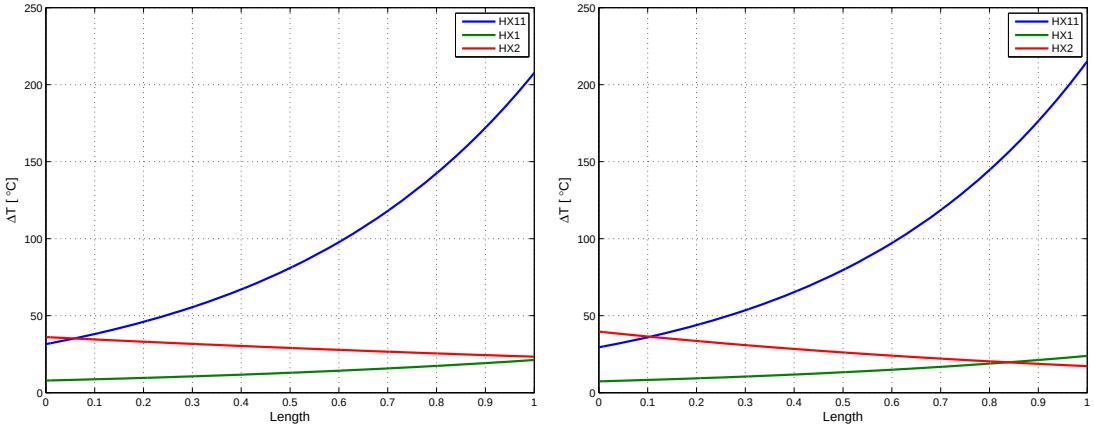
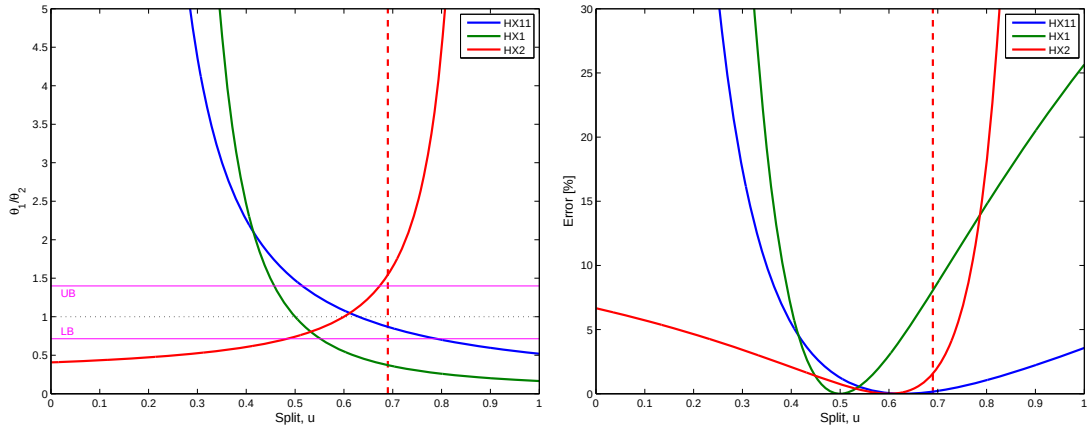


Figure B.1: T and c vs split



(a) Optimal (b) Self-optimizing

Figure B.2: Temperature profiles



(a) Approximation (b) Error

Figure B.3: Approximation

B.2 Case III-b: $\omega_h = \omega_c$ and low UA-values

	ω_h [W/K]	$T_{h,in}$ [°C]	UA [W/K]
HX11	25	70	30
HX1	50	160	70
HX2	25	125	33

Table B.2: Parameters for Case III-b

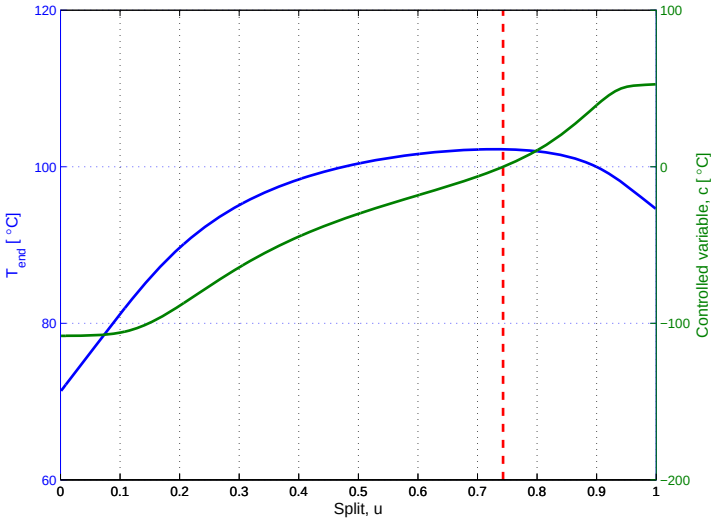
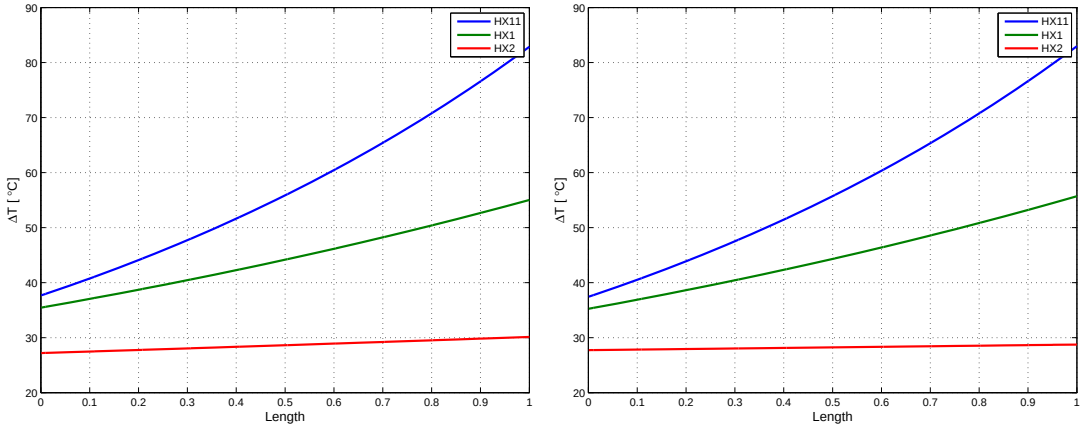
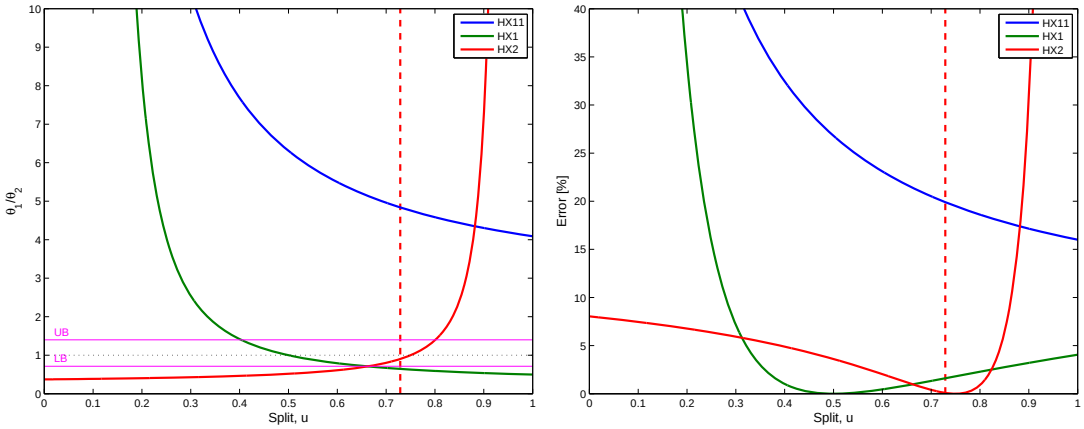


Figure B.4: T and c vs split



(a) Optimal (b) Self-optimizing

Figure B.5: Temperature profiles



(a) Approximation (b) Error

Figure B.6: Approximation

B.3 Case IV-b: $\omega_h > \omega_c$ and high UA-values

	ω_h [W/K]	$T_{h,in}$ [°C]	UA [W/K]
HX11	150	140	90
HX1	120	120	150
HX2	70	141	65

Table B.3: Parameters for Case IV-b

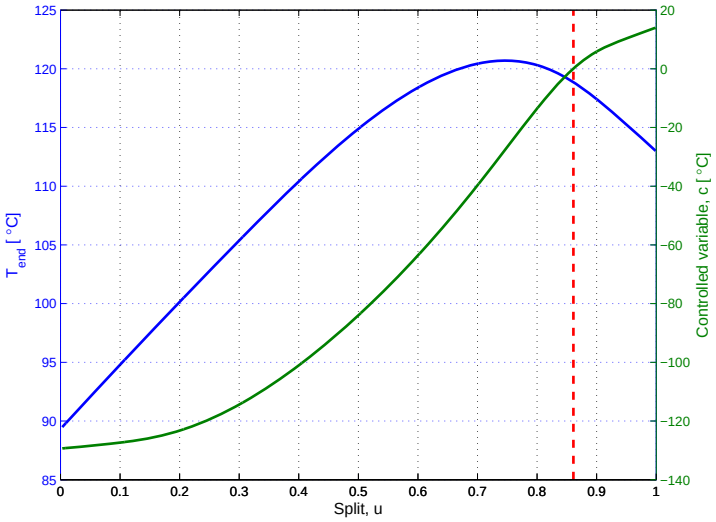
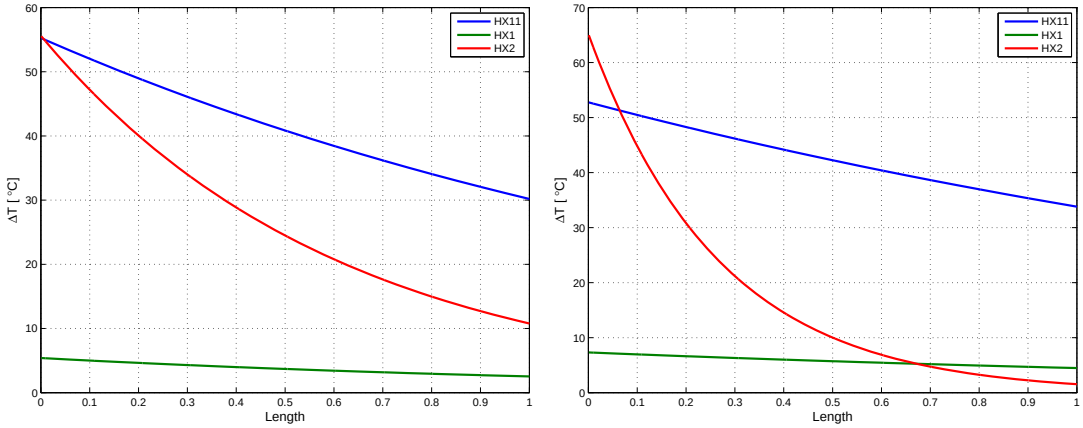
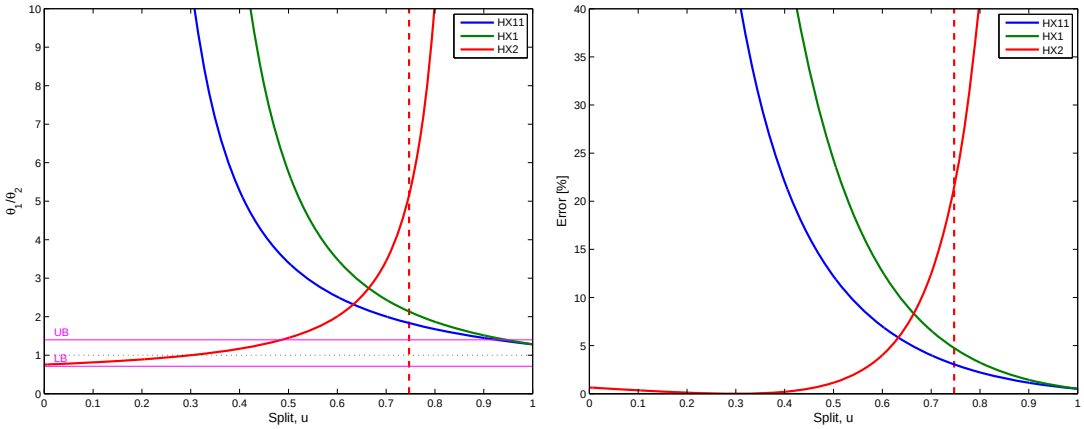


Figure B.7: T and c vs split



(a) Optimal (b) Self-optimizing

Figure B.8: Temperature profiles



(a) Approximation (b) Error

Figure B.9: Approximation

B.4 Case V-b: $\omega_h > \omega_c$ and intermediate UA-values

	ω_h [W/K]	$T_{h,in}$ [°C]	UA [W/K]
HX11	30	110	50
HX1	70	75	80
HX2	40	125	65

Table B.4: Parameters for Case V-b

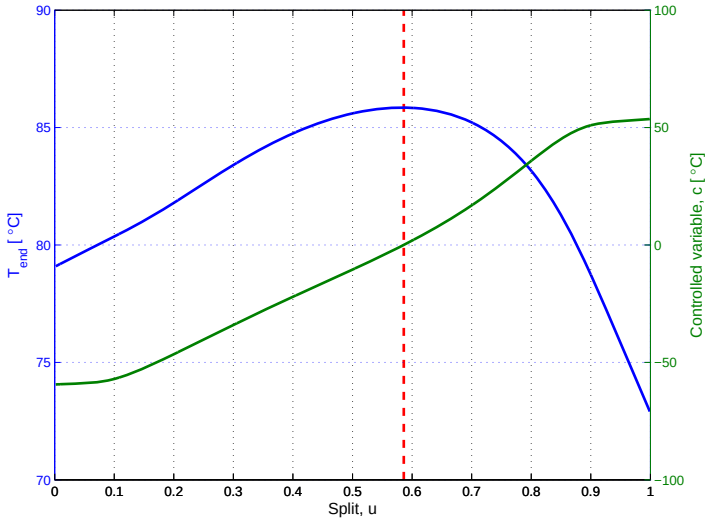
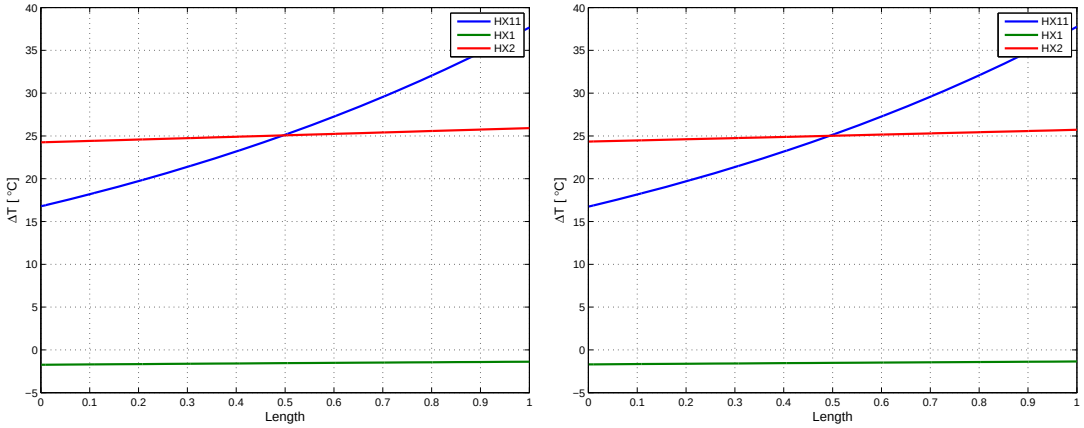
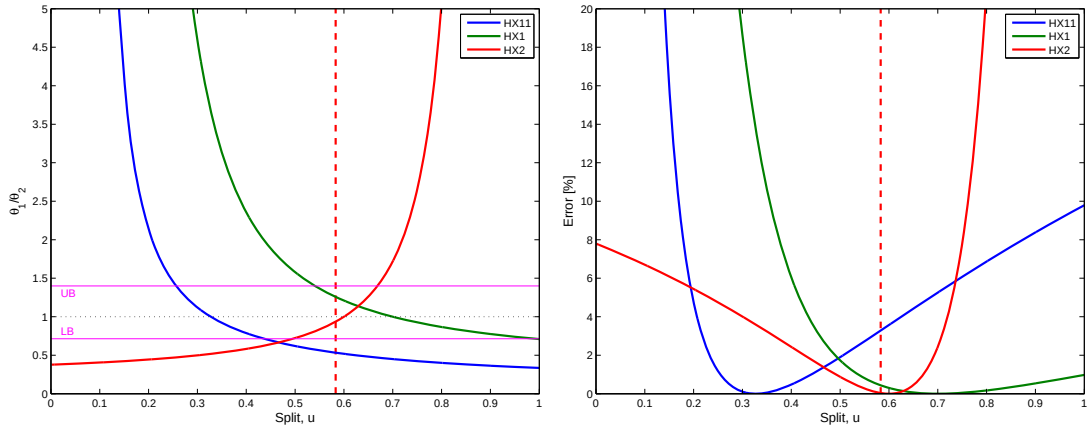


Figure B.10: T and c vs split



(a) Optimal (b) Self-optimizing

Figure B.11: Temperature profiles



(a) Approximation (b) Error

Figure B.12: Approximation

B.5 Case VI-b: $\omega_h > \omega_c$ and intermediate UA-values

	ω_h [W/K]	$T_{h,in}$ [°C]	UA [W/K]
HX11	30	120	50
HX1	50	140	80
HX2	40	80	65

Table B.5: Parameters for Case VI-b

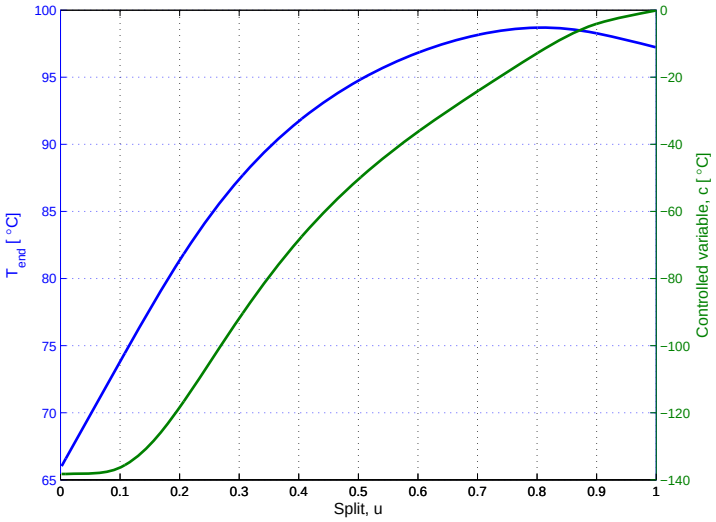


Figure B.13: T and c vs split

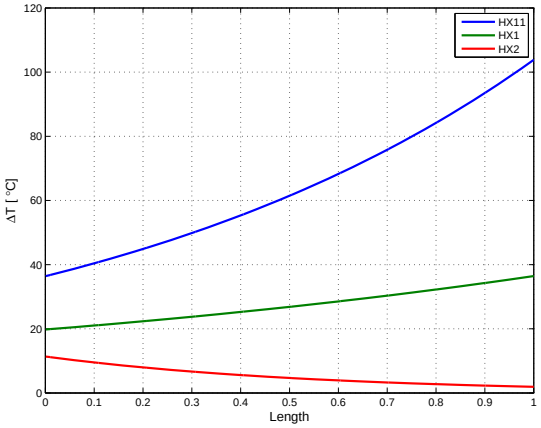
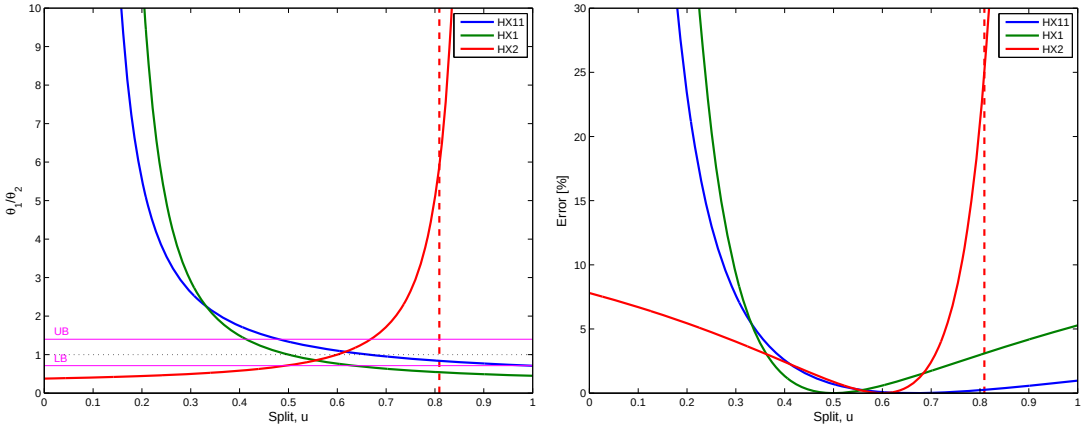


Figure B.14: Optimal temperature profile



(a) Approximation (b) Error

Figure B.15: Approximation

C Perstorp Flowsheet

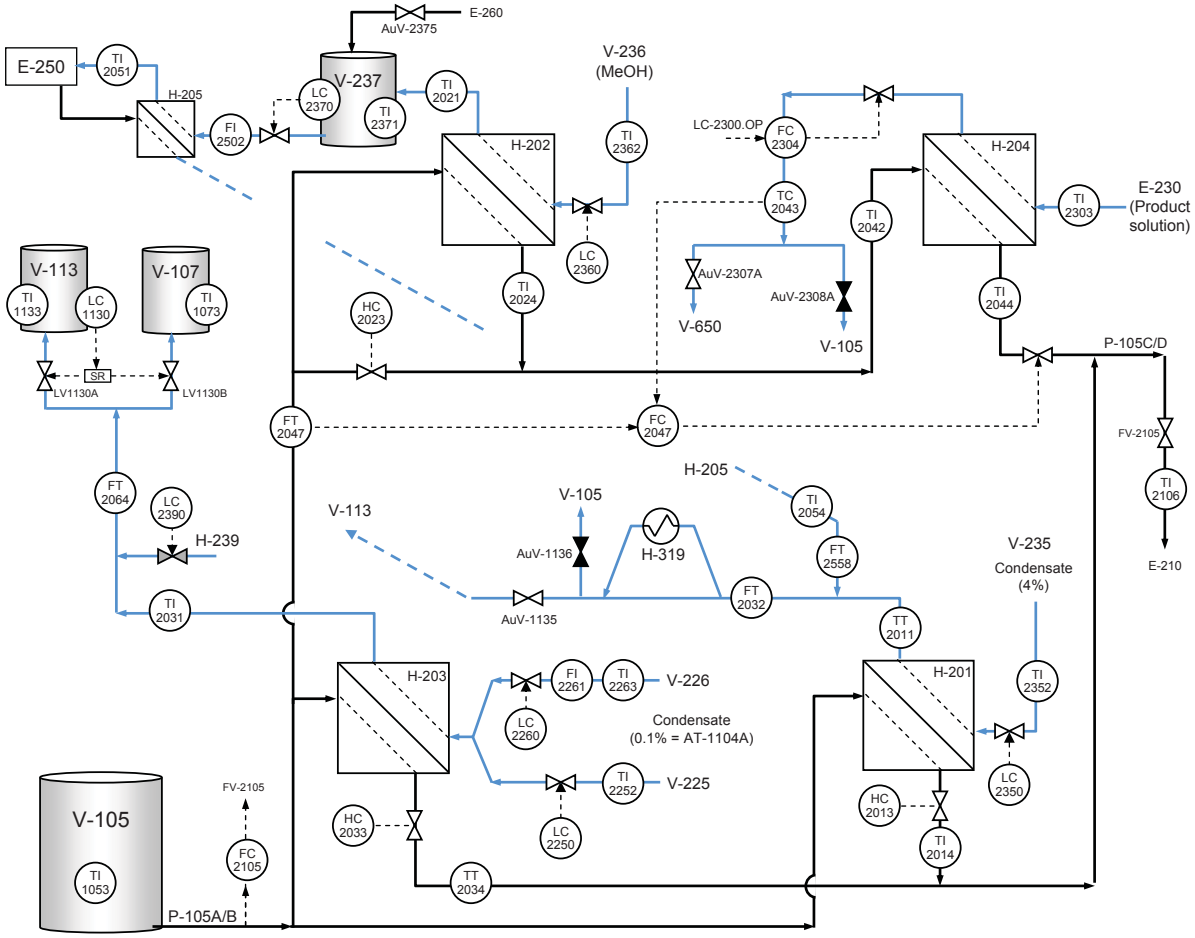


Figure C.1: Perstorp flowsheet

D Perstorp Case: Steady-State Results

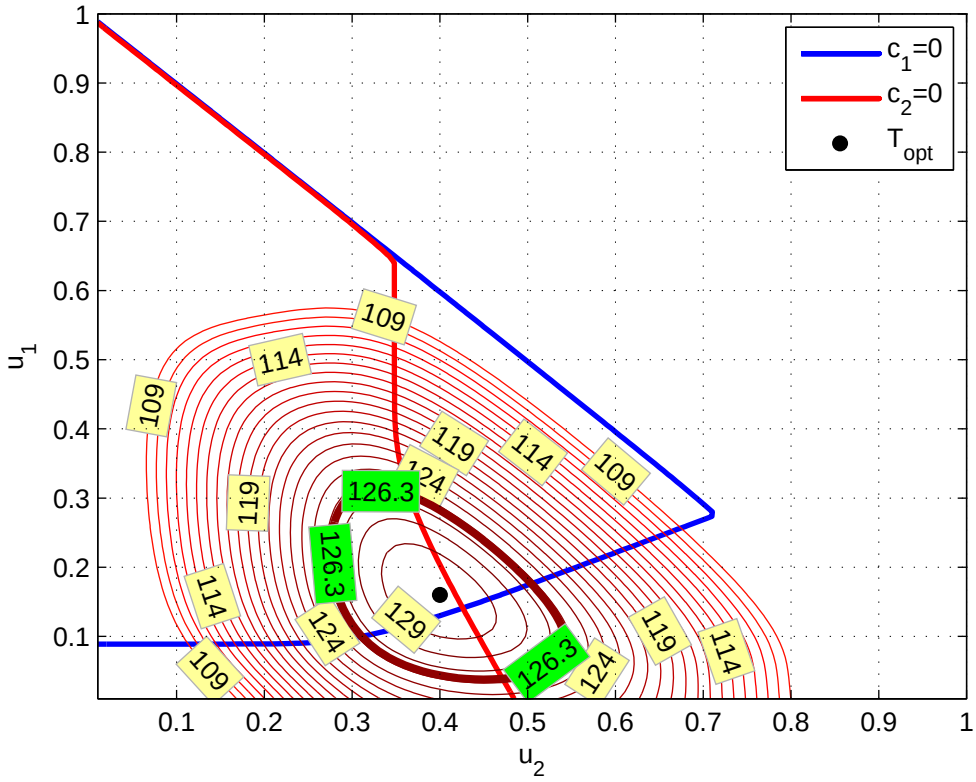


Figure D.1: Steady-state results (2008, January 12, 6pm)

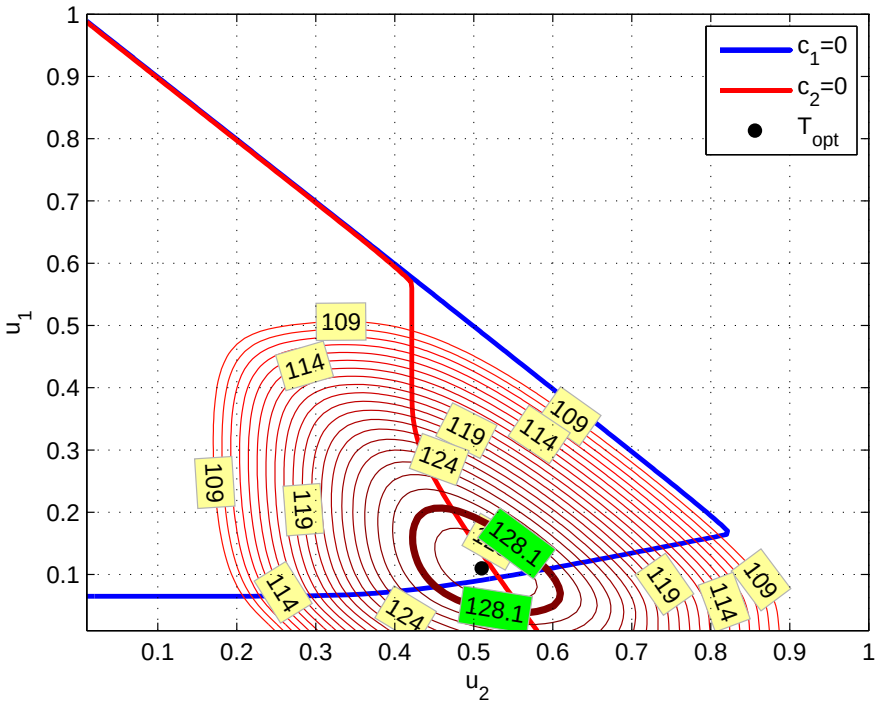


Figure D.2: Steady-state results (2008, September 27, 8pm)

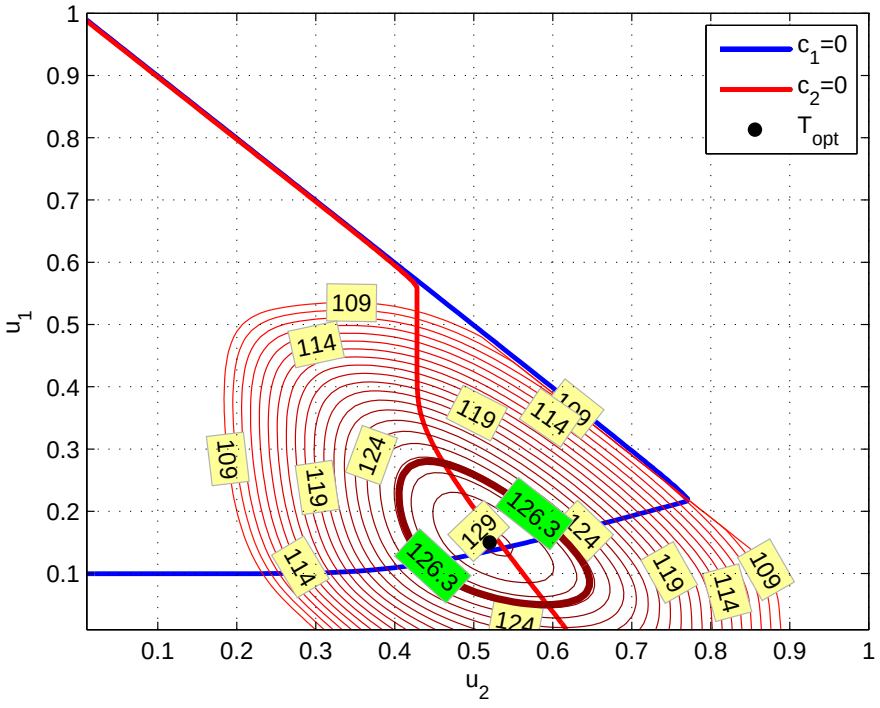


Figure D.3: Steady-state results (2008, December 14, 8pm)

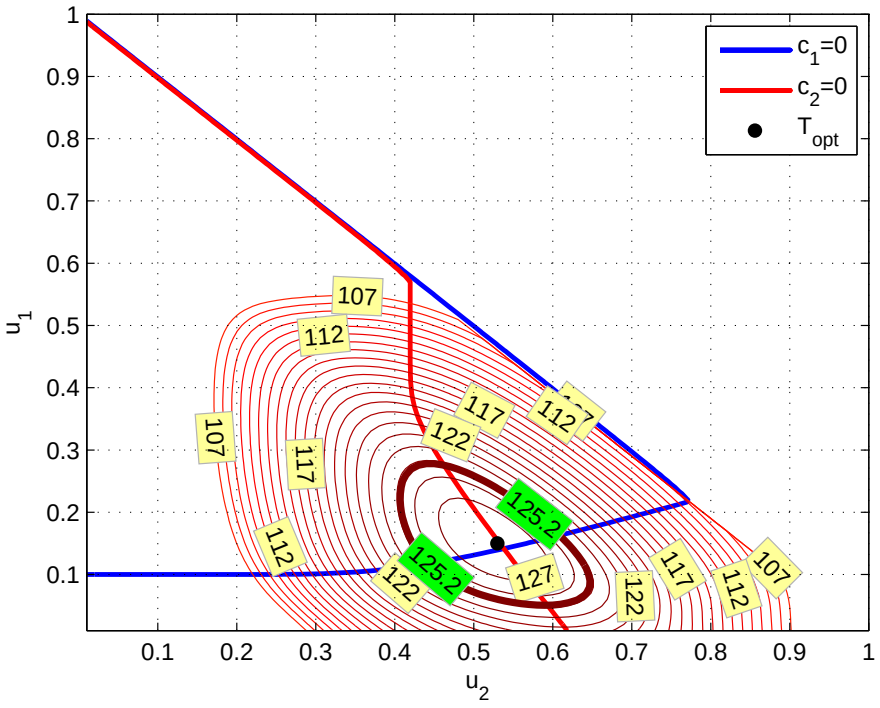


Figure D.4: Steady-state results (2009, February 22, 2pm)

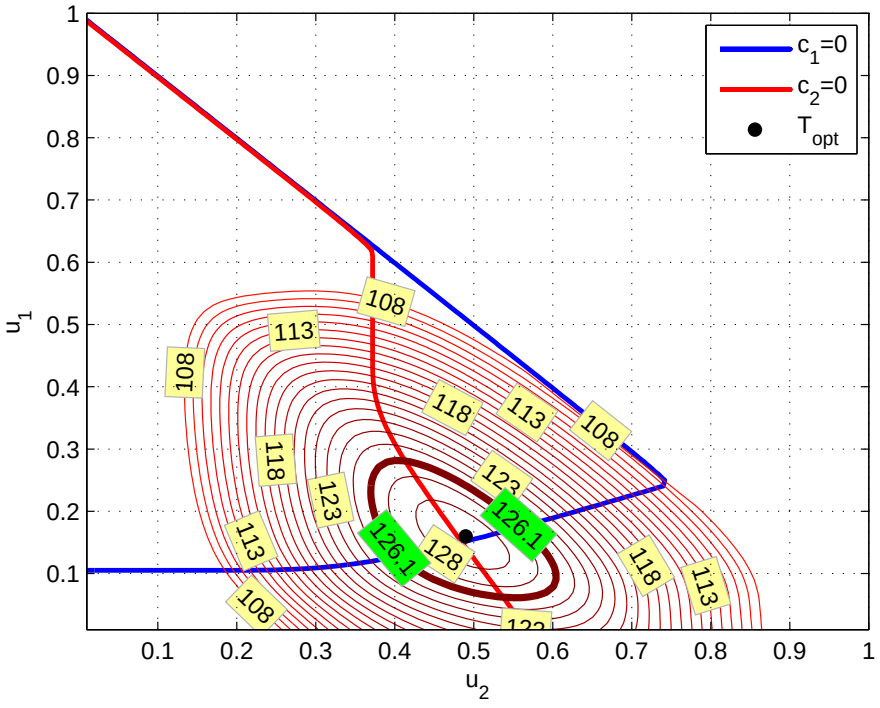


Figure D.5: Steady-state results (2009, September 18, 4pm)

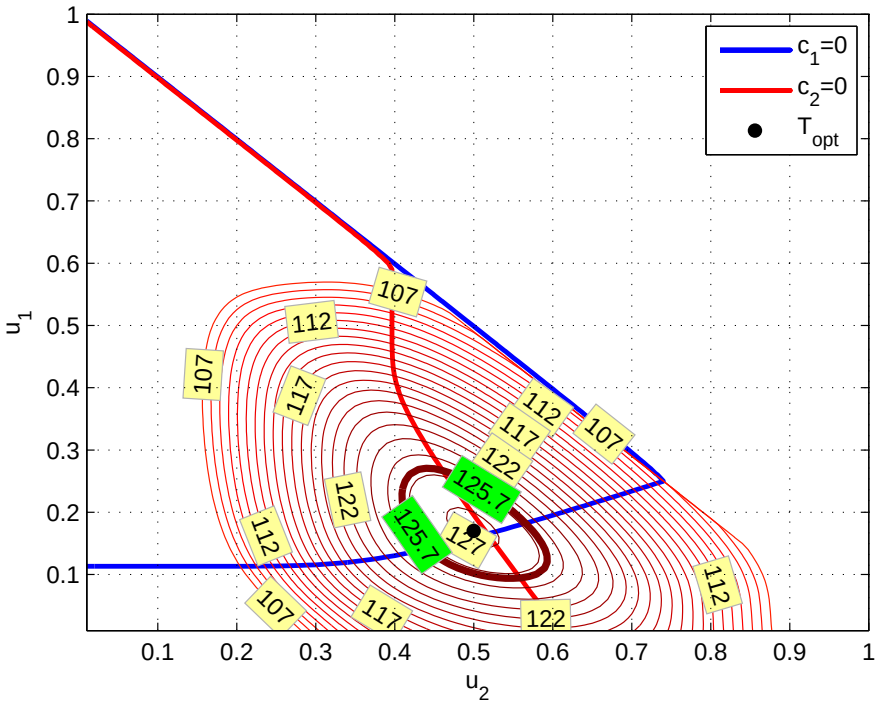


Figure D.6: Steady-state results (2009, December 17, 10pm)

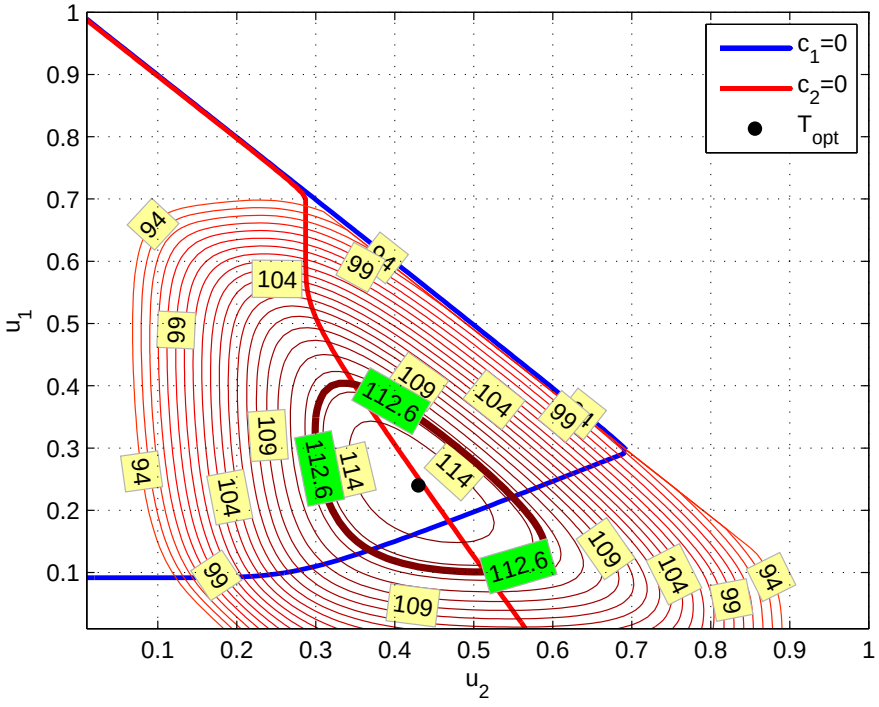


Figure D.7: Steady-state results (2010, February 21, 4pm)

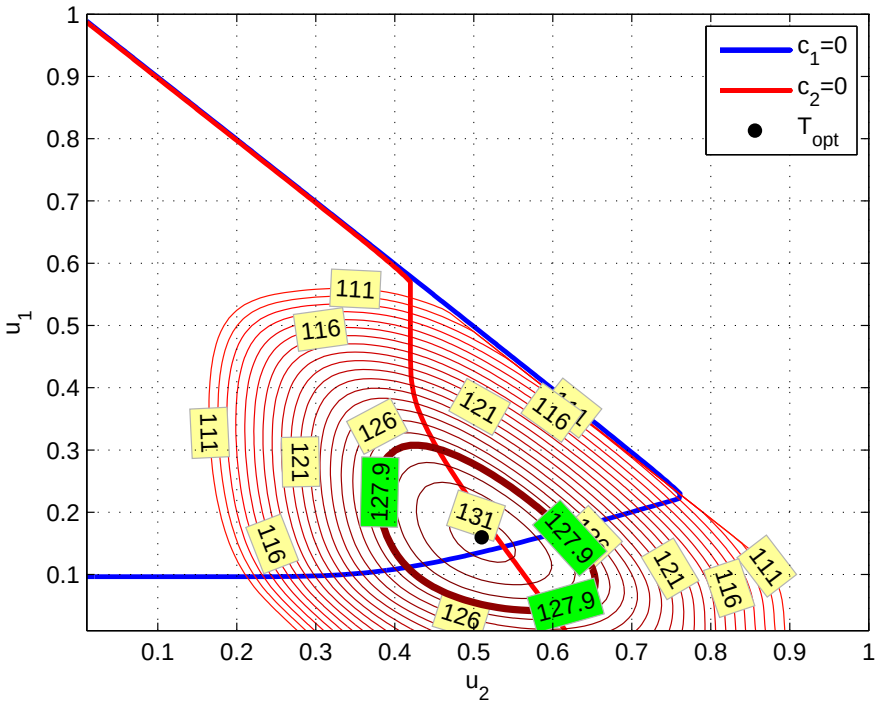


Figure D.8: Steady-state results (2010, August 15, 7pm)

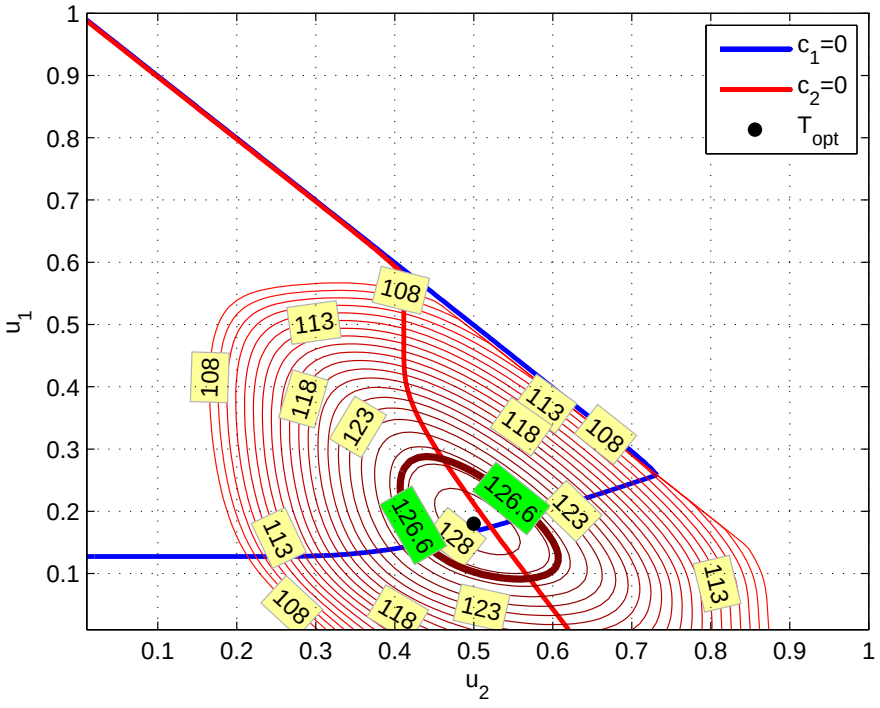


Figure D.9: Steady-state results (2010, December 11, 12pm)

E Mongstad HEN

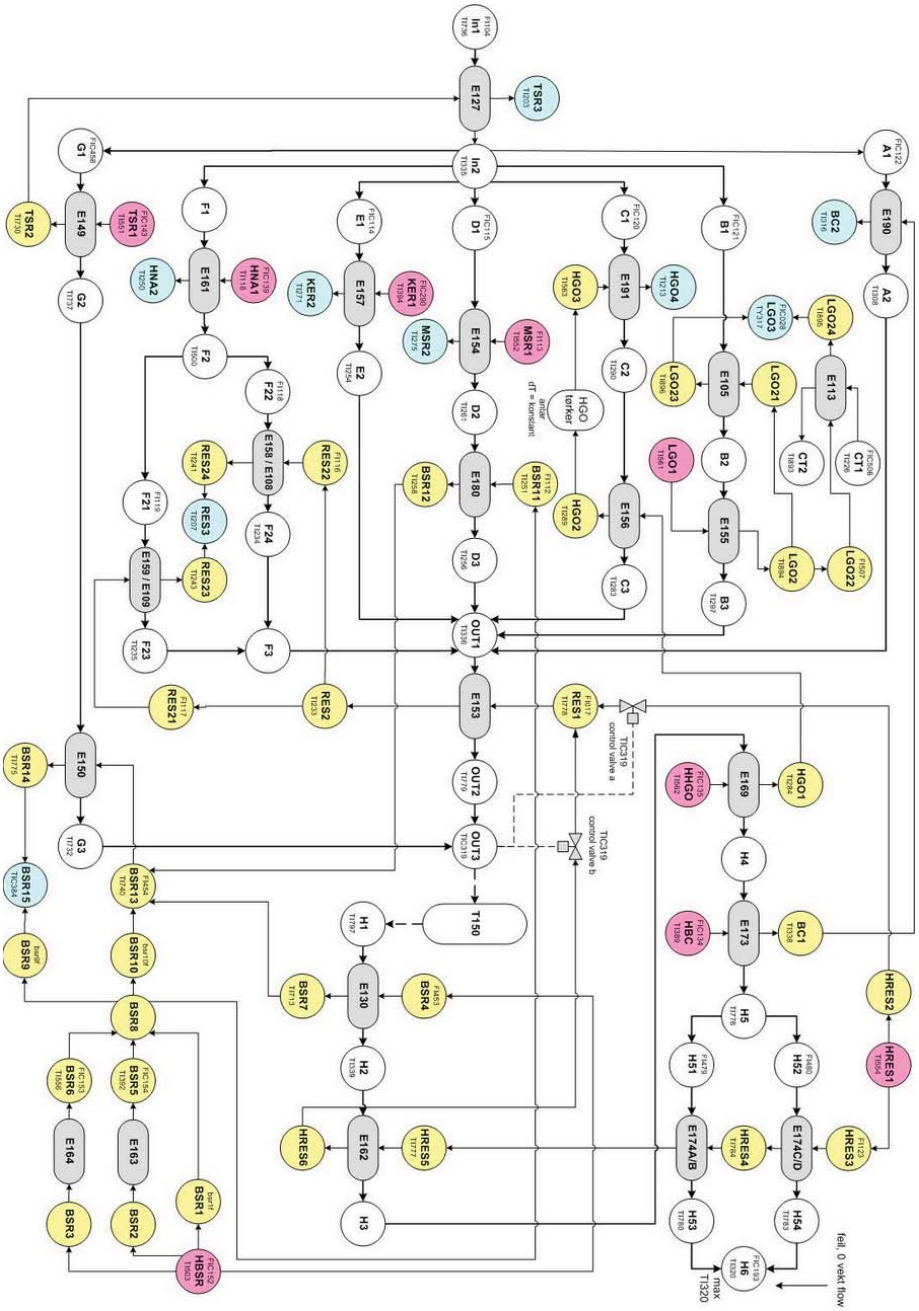


Figure E.1: Flowsheet of Mongstad HEN

F UA-values for Statoil Mongstad

	UA_{old} $\left[\frac{W}{K}\right]$	UA_{est} $\left[\frac{W}{K}\right]$	Difference $[\times 10^3\%]$
A1	131025.48	131023.97	-1.15
B1	102709.10	102710.93	+1.78
B2	88644.82	88645.03	+0.23
C1	84642.10	84641.36	-0.87
C2	133605.46	133605.42	-0.03
D1	132831.92	132831.76	-0.12
D2	41563.90	41564.42	+1.26
E1	190981.94	190981.21	-0.38
F1	49437.00	49438.07	+2.15
F2	111426.08	111425.97	-0.09
F3	112627.18	112627.77	+0.53
G1	96449.75	96445.64	-4.26
G2	125079.36	125075.55	-3.05
O1	144051.68	144050.14	-1.07

Table F.1: UA values

G MATLAB files

G.1 Case Studies

case1.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%                                                                 %%%%%
%%%%%                          CASE I                                 %%%%%
%%%%%       Two Heat Exchangers in Parallel                           %%%%%
%%%%%                                                                 %%%%%
%%%%%             Daniel Greiner Edvardsen                            %%%%%
%%%%%                     IKP, NTNU                                   %%%%%
%%%%%                   June 14, 2011                                 %%%%%
%%%%%                                                                 %%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Info: This m-file calculates the hot and cold outlet
% temperatures for two heat exchangers in parallel
% for a given set of conditions. Further, the optimum
% is found (optimal split) that results in the
% maximum end temperature. The SOC split is also found
% and compared with the optimal split.

clc;
clear all;
close all;

%% Defining parameters

% Cases evaluated
% Vector parameters: [T0 w0 wh1 wh2 Th1in Th2in UA1 UA2]

caseI    = [60 100  30 50  120 140  50 80];
caseII   = [60 100  30 50  120 140  120 200];
caseIII  = [60 100  15 15  120 140  120 200];
```

```
caseIV   = [60 100  50 50  120 140  50 80];
caseV    = [60 100  50 50  120 140  120 200];
caseVI   = [60 100  80 60  120 140  50 80];
caseVII  = [60 100  80 60  120 140  120 200];
caseVIII = [60 100  80 60  120 140  95 110];
HEN_F    = [147.0776 154477.15 91650.28 90371.74 ...
             248.07 248.07 110709 112203];
HEN_H    = [241.1040+0.8019 422096.96 159606.70 ...
             147970.52 358.5740 138842 209356];

% Select case
casesel = caseIII;

% Operation parameters
T0      = casesel(1); % Feed stream temperature [*C]
w0      = casesel(2); % [kW/K]

% Utility parameters
wh1     = casesel(3); % Hot stream 1 Heat Capacity [kW/K]
wh2     = casesel(4); % Hot stream 2 Heat Capacity [kW/K]
Th1in   = casesel(5); % Hot stream 1 Temperature
Th2in   = casesel(6); % Hot stream 2 Temperature

% Design parameters
UA1     = casesel(7);       % [kW/K]
UA2     = casesel(8);       % [kW/K]

% Number of iterations
iter=10000;

n = zeros(iter,1);
T1=n; T2=n; Th1=n; Th2=n; Tmix=n; e1=n; eh1=n; e2=n; eh2=n;
C1=n; C2=n; NTU1=n; NTU2=n; U=n;

%% T-out, HE and CONSTRAINT
for i=1:iter
```

```
67
68      u = i/iter;
69      U(i)=u;
70
71      % Calculating outlet temperatures and info about HEs
72      % (only u is changing)
73      [T HE] = tempCalc(T0,w0,UA1,UA2,Th1in,wh1,Th2in,wh2,u);
74
75      T1(i)=T(1); T2(i)=T(2); Th1(i)=T(3); Th2(i)=T(4);
76      Tmix(i)=T(5); e1(i)=HE(1); eh1(i)=HE(2); e2(i)=HE(3);
77      eh2(i)=HE(4); C1(i)=HE(5); C2(i)=HE(6); NTU1(i)=HE(7);
78      NTU2(i)=HE(8);
79
80  end
81
82  %% Results
83
84  % Self-optimizing variable
85  c = (T1-T0).^2./(Th1in-T0) - (T2-T0).^2./(Th2in-T0);
86
87  % Chen approximation
88  CHEN =  2.*T1.^2.*T2.^2.*Th2.*Th1in.*Th2in+8.*T1.*T2.^3.*
      Th2.*Th1.*Th1in-4.*T1.*Th2.*Th2in.*Th1in.^2.*T2.^2-4.*T1
      .*Th2.^2.*Th1.*Th1in.*T2.^2+4.*T1.*T2.^3.*Th2.*Th1in
      .^2+8.*T2.^2.*Th1.*Th2.*Th2in.*Th1in.^2-8.*T1.^2.*Th2.*
      Th1.*Th1in.*Th2in.^2-2.*T1.^2.*Th2.^2.*Th1.*Th2in.*Th1in
      +4.*T1.^2.*T2.*Th2.*Th1.^2.*Th2in+4.*T1.^2.*T2.*Th1.*
      Th1in.*Th2in.^2-2.*T1.^2.*T2.^2.*Th1.*Th2in.*Th1in+2.*T1
      .^3.*Th1.*Th2in.*T2.^2-4.*T1.^3.*T2.*Th1.*Th2in.^2+2.*T1
      .^3.*Th1.*Th2.^2.*Th2in+8.*T1.^3.*Th1.*Th2.*Th2in.^2-4.*
      T1.^2.*Th2.*Th1.^2.*Th2in.^2+2.*T1.^2.*T2.*Th1.^2.*Th2in
      .^2-T1.^2.*T2.^2.*Th1.^2.*Th2in+T1.^2.*T2.^2.*Th2.^2.*
      Th1in-2.*T1.^2.*Th2.*Th1in.*T2.^3-T1.^2.*Th2.^2.*Th1
      .^2.*Th2in-2.*T1.^2.*Th1.*Th2in.^3.*Th1in-2.*T1.*Th2
      .^2.*Th1in.^2.*T2.^2-2.*T2.^3.*Th1.^2.*Th2.*Th1in-8.*T2
      .^3.*Th1.*Th2.*Th1in.^2+T2.^2.*Th1.^2.*Th2.^2.*Th1in+4.*
```

```
    T2.^2.*Th1.*Th2.^2.*Th1in.^2+2.*T2.^2.*Th2.*Th1in.^3.*
    Th2in-T1.^2.*Th1.^2.*Th2in.^3+2.*T2.^2.*Th1.^2.*Th2.*
    Th1in.*Th2in-2.*T2.^3.*Th2.*Th1in.^3+T2.^2.*Th2.^2.*
    Th1in.^3+2.*T1.^3.*Th1.*Th2in.^3-8.*T1.^3.*T2.*Th1.*Th2
    .*Th2in+8.*T1.^2.*T2.*Th2.*Th1.*Th2in.*Th1in-8.*T1.*Th2
    .*Th1.*Th2in.*Th1in.*T2.^2;

%No exergy loss: T1 = T2
EQ = T1-T2;

% Gradient, GRAD
w1=w0.*U;
w2=w0.*(1.-U);

GRAD = UA2.*UA1.*T1.*wh2.*wh1+UA1.*w2.*T1.*UA2.*wh1+UA2.*T0
    .*w1.*wh2.*UA1-UA1.*T0.*w2.*UA2.*wh1-UA2.*UA1.*T2.*wh2.*
    wh1-UA2.*T2.*wh2.*w1.*UA1+2.*UA2.*T0.*w1.*wh2.*wh1-2.*
    UA2.*T2.*wh2.*w1.*wh1-2.*UA1.*T0.*w2.*wh2.*wh1+2.*UA1.*
    w2.*T1.*wh2.*wh1;

SUM = [U T1 Th1 T2 Th2 Tmix c];
HE = [e1 eh1 e2 eh2 C1 C2 NTU1 NTU2];

% Finding optimal split
[Tmixm,nr]=max(Tmix);

split=U(nr);
T1m=T1(nr);
Th1m=Th1(nr);
T2m=T2(nr);
Th2m=Th2(nr);
Ceq=c(nr);
Tmixm;

% Information about the heat transfer (optimal split)
he=[e1(nr) eh1(nr) e2(nr) eh2(nr) C1(nr) C2(nr) ...
```

```
115     NTU1(nr) NTU2(nr)]';
116
117 % Finding the self-optimizing split
118 [Ceq2,nr2]=min(abs(c));
119
120 split2=U(nr2);
121 T1m2=T1(nr2);
122 Th1m2=Th1(nr2);
123 T2m2=T2(nr2);
124 Th2m2=Th2(nr2);
125 Ceq2;
126 Tmixm2=Tmix(nr2);
127
128 % Information about the heat transfer
129 he2=[e1(nr2) eh1(nr2) e2(nr2) eh2(nr2) C1(nr2) C2(nr2) ...
130     NTU1(nr2) NTU2(nr2)]';
131
132 % Looking for the 'Gradient split'
133 [Ceq3,nr3]=min(abs(GRAD));
134 split3=U(nr3);
135
136 % Looking for the 'T1-T2 split'
137 [Ceq4,nr4]=min(abs(EQ));
138 split4=U(nr4);
139
140 % Looking for the 'CHEN split'
141 [Ceq5,nr5]=min(abs(CHEN));
142 split5=U(nr5);
143
144 % Deviation from optimal split
145 best = abs([split2,split3,split4,split5]'-split);
146
147 % Calculating the temperature differences on each side of
148 % the HEs
149 dT11=Th1-T0; dT12=Th1in-T1; dT21=Th2-T0; dT22=Th2in-T2;
150
```

```
151 Tr1=dT11./dT12;
152 Tr2=dT21./dT22;
153
154 % Calculating errors
155 [eAM1 eAM2] = errCalc(dT11, dT12, dT21, dT22);
156
157 %% Some plots
158 close all;
159
160 % Tend and controlled variable
161 h = figure;
162
163 y1start=190; y1slutt=212; sprangy1=2;
164 y2start=-100; y2slutt=120; sprangy2=20;
165
166 [AX,H1,H2] = plotyy(U, Tmix, U,c);
167 set(get(AX(1),'Ylabel'),'String',...
168     'T_{end} [ \circC]','fontsize',12)
169 set(get(AX(2),'Ylabel'),'String',...
170     'Controlled variable, c [ \circC]','fontsize',12)
171 axis(AX(1),[0 1 y1start y1slutt]);
172 axis(AX(2),[0 1 y2start y2slutt]);
173 set(AX(1),'YLim',[y1start y1slutt])
174 set(AX(1),'YTick',y1start:sprangy1:y1slutt)
175 set(AX(2),'YLim',[y2start y2slutt])
176 set(AX(2),'YTick',y2start:sprangy2:y2slutt)
177 set(H1,'linewidth',2)
178 set(H2,'linewidth',2)
179 xlabel('Split, u','fontsize',12);
180 hold on;
181 gridxy(split2,'Color','r','LineStyle','--','LineWidth',2);
182 hold on;
183 grid on;
184 print(h,'-depsc','temp.eps');
185
186 %Error plots
```

```
187 h = figure;
188 plot(U,Tr1,U,Tr2,'linewidth',2);
189 axis([0 1 0 0.1]);
190 xlabel('Split, u','fontsize',12);
191 ylabel('\theta_1/\theta_2','fontsize',12);
192 legend('HX1','HX2','fontsize',12);
193 hline([1/1.4 1.4 1],{'magenta','magenta','black:'},...
194     {'LB','UB',''})
195 hold on;
196 gridxy(split,'Color','r','LineStyle','--','LineWidth',2);
197 print(h,'-depsc','approx.eps');
198
199 h = figure;
200 plot(U,eAM1,U,eAM2,'linewidth',2);
201 xlabel('Split, u','fontsize',12);
202 ylabel('Error [%]','fontsize',12);
203 legend('HX1','HX2','fontsize',12);
204 axis([0 1 0 500]);
205 hold on;
206 gridxy(split,'Color','r','LineStyle','--','LineWidth',2);
207 print(h,'-depsc','error.eps');
208
209 % Temperature profiles
210
211 %Optimal profiles
212 w1opt=split*w0;
213 w2opt=(1-split)*w0;
214
215 [z dT1opt dT2opt] = Tprof(T0,Th1m,Th2m,w1opt,w2opt,...
216     wh1,wh2,UA1,UA2);
217
218 %SOC profiles
219 w1soc=split2*w0;
220 w2soc=(1-split2)*w0;
221
222 [z dT1soc dT2soc] = Tprof(T0,Th1m2,Th2m2,w1soc,w2soc,...
```

```
223      wh1,wh2,UA1,UA2);
224
225  %Optimal
226  h = figure;
227  plot(z,dT1opt,z,dT2opt,'linewidth',2);
228  xlabel('Length','fontsize',12);
229  ylabel('\DeltaT [ \circC]','fontsize',12);
230  legend('HX1','HX2','fontsize',12);
231  grid on;
232  print(h,'-depsc','optprof.eps');
233
234  %SOC
235  h = figure;
236  plot(z,dT1soc,z,dT2soc,'linewidth',2);
237  xlabel('Length','fontsize',12);
238  ylabel('\DeltaT [ \circC]','fontsize',12);
239  legend('HX1','HX2','fontsize',12);
240  grid on;
241  print(h,'-depsc','socprof.eps');
```

tempCalc.m

```
1  function [T HE] = tempCalc(T0,w0,UA1,UA2,Th1in,wh1,...
2                             Th2in,wh2,u)
3
4  w1 = u*w0;
5  w2 = (1-u)*w0;
6
7  NTU1 = UA1/w1;
8  NTU2 = UA2/w2;
9
10 C1 = w1/wh1;
11 C2 = w2/wh2;
12
13 if(C1>0.999 && C1<1.001)
14     C1=0.999;
```

```
15 end
16
17 if(C2>0.999 && C2<1.001)
18     C2=0.999;
19 end
20
21 e1 = (1-exp(-NTU1*(C1-1)))/(C1-exp(-NTU1*(C1-1)));
22 e2 = (1-exp(-NTU2*(C2-1)))/(C2-exp(-NTU2*(C2-1)));
23 eh1 = e1*C1;
24 eh2 = e2*C2;
25
26 T1  = e1*Th1in + (1-e1)*T0;
27 T2  = e2*Th2in + (1-e2)*T0;
28 Th1 = (1-eh1)*Th1in + eh1*T0;
29 Th2 = (1-eh2)*Th2in + eh2*T0;
30 Tmix = u*T1+(1-u)*T2;
31
32 T = [T1 T2 Th1 Th2 Tmix];
33 HE = [e1 eh1 e2 eh2 C1 C2 NTU1 NTU2];
```

errCalc.m

```
1 function [eAM1 eAM2] = errCalc(dT11, dT12, dT21, dT22)
2
3 %Logarithmic mean temperature difference
4 LM1 = (dT11-dT12)./log(dT11./dT12);
5 LM2 = (dT21-dT22)./log(dT21./dT22);
6
7 %Arithmetic mean temperature difference
8 AM1 = (dT11+dT12)./2;
9 AM2 = (dT21+dT22)./2;
10
11 %AM error
12 eAM1 = (AM1-LM1)./LM1*100;
13 eAM2 = (AM2-LM2)./LM2*100;
```

Tprof.m

```
function [z dT1 dT2] = Tprof(T0,Th1,Th2,w1,w2,wh1,wh2,UA1,
    UA2)

z=(0.0001:0.0001:1)';

R1=(w1-wh1)./(w1*wh1);
R2=(w2-wh2)./(w2*wh2);

dT1 = (Th1-T0).*exp(R1*UA1.*z);
dT2 = (Th2-T0).*exp(R2*UA2.*z);
```

case2.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%                                                                 %%%%%
%%%%%                          CASE II                                %%%%%
%%%%%              Two Heat Exchangers in Series                      %%%%%
%%%%%                    and One in Parallel                          %%%%%
%%%%%                                                                 %%%%%
%%%%%                Daniel Greiner Edvardsen                         %%%%%
%%%%%                        IKP, NTNU                                %%%%%
%%%%%                      June 14, 2011                              %%%%%
%%%%%                                                                 %%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Info: This m-file calculates the hot and cold outlet
% temperatures for two heat exchangers in series and
% one in parallel for a given set of conditions.
% Further, the optimum is found (optimal split) that
% results in the maximum end temperature. The SOC split
% is also found and compared with the optimal split.

clc;
clear all;
close all;

%% Defining parameters

% Cases evaluated. Vector parameters:
% [T0 w0 wh11 wh1 wh2 Th11in Th1in Th2in UA11 UA1 UA2]
caseI    = [60 100  30 50 20   120 140 140  50 80 65];
caseII   = [60 100  30 50 40   120 140 140  100 180 60];
caseIII  = [60 100  25 50 25   70 160 125   30 70 33];
caseIV   = [60 100  150 120 70 140 120 141  90 150 65];
caseV    = [60 100  30 70 40   110 75 125   50 80 65];
caseVI   = [60 100  30 50 40   120 140 80   50 80 65];

%Select case
```

```
casesel = caseI;

%%%%%%%%%%%%%%% Operation parameters %%%%%%%%%%%%%%%%%%%
T0  = casesel(1);       % Feed stream temperature [K]
w0  = casesel(2);       % m*Cp [kW/K]
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%% Utility parameters %%%%%%%%%%%%%%%%%%%%%
wh11    = casesel(3); % Hot stream 1 Heat Capacity  [kW/K]
wh1     = casesel(4); % Hot stream 11 Heat Capacity [kW/K]
wh2     = casesel(5); % Hot stream 2 Heat Capacity  [kW/K]
Th11in  = casesel(6); % Hot stream 1 Temperature    [kW/K]
Th1in   = casesel(7); % Hot stream 11 Temperature   [kW/K]
Th2in   = casesel(8); % Hot stream 2 Temperature    [kW/K]
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%% Design parameters %%%%%%%%%%%%%%%%%%%
UA11    = casesel(9);   % [kW/K]
UA1     = casesel(10);  % [kW/K]
UA2     = casesel(11);  % [kW/K]
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Number of iterations
iter=1000;

n = zeros(iter,1);
T11=n; T2=n; T1=n; Th11=n; Th2=n; Th1=n; Tmix=n;
e11=n; eh11=n; e2=n; eh2=n; e1=n; eh1=n; C11=n;
C2=n; C1=n; NTU11=n; NTU2=n; NTU1=n; U=n; CON=n;
CHEN=n; CHEN2=n; CON2=n;

%% T-out, HE and CONSTRAINT
for i=1:iter

    u = i/iter;
    U(i)=u;
```

```
72
73      % Calculating outlet temperatures and info about HEs
74      % Only u is changing
75      [T HE] = tempCalc2(T0,w0,UA11,UA2,UA1,Th11in,Th2in,...
76          Th1in,wh11,wh2,wh1,u);
77
78      T11(i)=T(1); T2(i)=T(2); T1(i)=T(3); Th11(i)=T(4);
79      Th2(i)=T(5); Th1(i)=T(6); Tmix(i)=T(7); e11(i)=HE(1);
80      eh11(i)=HE(2); e2(i)=HE(3); eh2(i)=HE(4); C11(i)=HE(5);
81      C2(i)=HE(6); NTU11(i)=HE(7); NTU2(i)=HE(8);
82
83  end
84
85  %% Results
86
87  % Self-optimizing variable
88  c = (T2-T0).^2./(Th2in-T0) - (((Th1in-T1)./(Th11in-T0)-1)
        .*(T11-T0).^2./(Th1in-T11)+(T1-T0).^2./(Th1in-T11));
89
90  %No exergy loss: T1 = T2
91  EQ = T1-T2;
92
93  % Gradient, GRAD
94  w1=w0.*U;
95  w2=w0.*(1.-U);
96
97  GRAD = -4.*UA2.*T2.*wh1.*w1.^2.*wh11.*wh2-2.*UA2.*T2.*w1
        .^2.*UA1.*wh11.*wh2+2.*w1.*T1.*wh1.*UA1.*wh11.*w2.*UA2
        +4.*w1.*T1.*wh1.*UA1.*wh11.*w2.*wh2+2.*T1.*UA1.*wh1.*w1
        .*wh11.*w2.*UA2+4.*T1.*UA1.*wh1.*w1.*wh11.*w2.*wh2+2.*
        UA2.*w1.*T1.*wh1.*UA1.*wh11.*wh2+2.*UA2.*T1.*UA1.*wh1.*
        w1.*wh11.*wh2-2.*UA2.*T2.*w1.*wh1.*UA1.*wh11.*wh2-2.*UA2
        .*UA11.*T2.*wh1.*w1.^2.*wh2-UA2.*T2.*w1.^2.*UA1.*UA11.*
        wh2+2.*UA11.*w1.*T1.*wh1.*wh11.*w2.*UA2+4.*UA11.*w1.*T1
        .*wh1.*wh11.*w2.*wh2+2.*UA2.*UA11.*w1.*T1.*wh1.*wh11.*
        wh2-2.*UA2.*UA11.*T2.*w1.*wh1.*wh11.*wh2+UA1.*UA11.*T1.*
```

```
    wh1.*wh11.*w2.*UA2+UA11.*w1.*T1.*wh1.*UA1.*w2.*UA2+2.*
    UA11.*T1.*UA1.*wh1.*w1.*w2.*wh2+UA11.*T1.*UA1.*wh1.*w1.*
    w2.*UA2+2.*UA11.*w1.*T1.*wh1.*UA1.*w2.*wh2+UA2.*UA11.*w1
    .*T1.*wh1.*UA1.*wh2+UA2.*UA11.*T1.*UA1.*wh1.*w1.*wh2-UA2
    .*UA11.*T2.*w1.*wh1.*UA1.*wh2+UA11.*w1.*T1.*UA1.*wh11.*
    w2.*UA2+UA2.*UA11.*w1.*T1.*UA1.*wh11.*wh2+2.*UA11.*w1.*
    T1.*UA1.*wh11.*w2.*wh2-UA2.*UA11.*T2.*w1.*UA1.*wh11.*wh2
    +2.*UA1.*UA11.*T1.*wh1.*wh11.*w2.*wh2+UA1.*UA2.*UA11.*T1
    .*wh1.*wh11.*wh2-UA1.*UA2.*UA11.*T2.*wh1.*wh11.*wh2;

SUM = [U T11 Th1 T2 Th2 Tmix c];
HE = [e11 eh11 e2 eh2 C11 C2 NTU11 NTU2];

% Finding optimal split
[Tmixm,nr]=max(Tmix);

split=U(nr);
T11m=T11(nr); T2m=T2(nr); T1m=T1(nr); Th11m=Th1(nr);
Th2m=Th2(nr); Th1m=Th1(nr); Ceq=c(nr); Tmixm;

Tm=[T11m T2m T1m Th11m Th2m Th1m Tmixm]';

% Information about the heat transfer (optimal)
he=[e11(nr) eh11(nr) e2(nr) eh2(nr) C11(nr) C2(nr)
    NTU11(nr) NTU2(nr)]';

% 'theta1'/'theta2 (in optimum)'
Tr=[(Th11m-T0)/(Th11in-T11m)
    (Th2m-T0)/(Th2in-T2m)
    (Th1m-T11m)/(Th1in-T1m)];

% Calculating the temperature differences
% on each side of the HEs
dT111=Th1-T0; dT112=Th11in-T11; dT21=Th2-T0;
dT22=Th2in-T2; dT11=Th1-T11; dT12=Th1in-T1;

```

```
125 Tr11=dT111./dT112;
126 Tr2=dT21./dT22;
127 Tr1=dT11./dT12;
128
129 % Calculating errors
130 [eAM11 eAM2 eAM1] = errCalc2(dT111, dT112, dT21, dT22, ...
131     dT11, dT12);
132
133 int=round(split*iter);
134 eTOT=sqrt(eAM11(int)^2+eAM2(int)^2+eAM1(int)^2);
135
136 % Finding the self-optimizing split
137 [Ceq2,nr2]=min(abs(c));
138 split2=U(nr2);
139 T11m2=T11(nr2); Th11m2=Th1(nr2); T2m2=T2(nr2);
140 Th2m2=Th2(nr2); T1m2=T1(nr2); Th1m2=Th1(nr2);
141 Ceq2; Tmixm2=Tmix(nr2);
142
143 Tm2=[T11m2 T2m2 T1m2 Th11m2 Th2m2 Th1m2 Tmixm2]';
144
145 % Information about the heat transfer (SCOC)
146 he2=[e11(nr2) eh11(nr2) e2(nr2) eh2(nr2) C11(nr2) C2(nr2)
147     NTU11(nr2) NTU2(nr2)]'
148
149 % Looking for the 'Gradient split'
150 [Ceq3,nr3]=min(abs(GRAD));
151 split3=U(nr3);
152
153 % Looking for the 'T11-T2 split'
154 [Ceq4,nr4]=min(abs(EQ));
155 split4=U(nr4);
156
157 % Looking for the 'CHEN split'
158 [Ceq5,nr5]=min(abs(CHEN));
159 split5=U(nr5);
160
```

```
% Deviation from optimal split
best = abs([split2,split3,split4,split5]'-split);

%% Plots
close all;

% Tend and controlled variable
h = figure;
[AX,H1,H2] = plotyy(U, Tmix, U,c);
set(get(AX(1),'Ylabel'),'String','T_{end} [ \circC]',...
    'fontsize',12)
set(get(AX(2),'Ylabel'),'String',...
    'Controlled variable, c [ \circC]','fontsize',12)
set(H1,'linewidth',2)
set(H2,'linewidth',2)
xlabel('Split, u','fontsize',12);
hold on;
gridxy(split2,'Color','r','LineStyle','--','LineWidth',2);
hold on;
grid on;
print(h,'-depsc','temp.eps');

%Error plots
h = figure;
plot(U,Tr11,U,Tr1,U,Tr2,'linewidth',2);
axis([0 1 0 10]);
xlabel('Split, u','fontsize',12);
ylabel('\theta_1/\theta_2','fontsize',12);
legend('HX11','HX1','HX2','fontsize',12);
hline([1/1.4 1.4 1],{'magenta','magenta','black:'},...
    {'LB','UB',''})
hold on;
gridxy(split,'Color','r','LineStyle','--','LineWidth',2);
print(h,'-depsc','approx.eps');

h = figure;
```

```
plot(U,eAM11,U,eAM1,U,eAM2,'linewidth',2);
xlabel('Split, u','fontsize',12);
ylabel('Error [%]','fontsize',12);
legend('HX11','HX1','HX2','fontsize',12);
axis([0 1 0 30]);
hold on;
gridxy(split,'Color','r','LineStyle','--','LineWidth',2);
print(h,'-depsc','error.eps');

% Temperature profiles

%Optimal profiles
w1opt=split*w0;
w2opt=(1-split)*w0;

[z dT11opt dT2opt dT1opt] = Tprof2(T0,T11m,Th11m,Th2m,...
    Th1m,w1opt,w2opt,wh11,wh2,wh1,UA11,UA2,UA1,iter);

%SOC profiles
w1soc=split2*w0;
w2soc=(1-split2)*w0;

[z dT11soc dT2soc dT1soc] = Tprof2(T0,T11m2,Th11m2,...
    Th2m2,Th1m2,w1soc,w2soc,wh11,wh2,wh1,UA11,...
    UA2,UA1,iter);

% Optimal
h = figure;
plot(z,dT11opt,z,dT1opt,z,dT2opt,'linewidth',2);
xlabel('Length','fontsize',12);
ylabel('\DeltaT [ \circC]','fontsize',12);
legend('HX11','HX1','HX2','fontsize',12);
grid on;
print(h,'-depsc','optprof.eps');

% SOC
```

```
233 h = figure;
234 plot(z,dT11soc,z,dT1soc,z,dT2soc,'linewidth',2);
235 xlabel('Length','fontsize',12);
236 ylabel('\DeltaT [ \circC]','fontsize',12);
237 legend('HX11','HX1','HX2','fontsize',12);
238 grid on;
239 print(h,'-depsc','socprof.eps');
```

tempCalc2.m

```
1  function [T HE] = tempCalc2(T0,w0,UA1,UA2,UA3,...
2      Th1in,Th2in,Th3in,wh1,wh2,wh3,u)
3
4  w1 = u*w0;
5  w2 = (1-u)*w0;
6
7  NTU1 = UA1/w1;
8  NTU2 = UA2/w2;
9  NTU3 = UA3/w1;
10
11 C1 = w1/wh1;
12 C2 = w2/wh2;
13 C3 = w1/wh3;
14
15 if(C1>0.99999 && C1<1.00001)
16     C1=0.99999;
17 end
18
19 if(C2>0.99999 && C2<1.00001)
20     C2=0.99999;
21 end
22
23 if(C3>0.99999 && C3<1.00001)
24     C3=0.99999;
25 end
26
```

```
e1 = (1-exp(-NTU1*(C1-1)))/(C1-exp(-NTU1*(C1-1)));
e2 = (1-exp(-NTU2*(C2-1)))/(C2-exp(-NTU2*(C2-1)));
e3 = (1-exp(-NTU3*(C3-1)))/(C3-exp(-NTU3*(C3-1)));

eh1 = e1*C1;
eh2 = e2*C2;
eh3 = e3*C3;

T1  = e1*Th1in + (1-e1)*T0;
T2  = e2*Th2in + (1-e2)*T0;
T3  = e3*Th3in + (1-e3)*T1;

Th1 = (1-eh1)*Th1in + eh1*T0;
Th2 = (1-eh2)*Th2in + eh2*T0;
Th3 = (1-eh3)*Th3in + eh3*T1;

Tmix = u*T3+(1-u)*T2;

T = [T1 T2 T3 Th1 Th2 Th3 Tmix];
HE = [e1 eh1 e2 eh2 e3 eh3 C1 C2 C3 NTU1 NTU2 NTU3];
```

errCalc2.m

```
function [eAM1 eAM2 eAM3] = errCalc2(dT11, dT12, dT21,...
    dT22, dT31, dT32)

% Logarithmic mean temperature difference
LM1 = (dT11-dT12)./log(dT11./dT12);
LM2 = (dT21-dT22)./log(dT21./dT22);
LM3 = (dT31-dT32)./log(dT31./dT32);

% Arithmetic mean temperature difference
AM1 = (dT11+dT12)./2;
AM2 = (dT21+dT22)./2;
AM3 = (dT31+dT32)./2;

```

```
%AM error
eAM1 = ((AM1-LM1)./LM1).*100;
eAM2 = ((AM2-LM2)./LM2).*100;
eAM3 = ((AM3-LM3)./LM3).*100;
```

Tprof2.m

```
function [z dT1 dT2 dT3] = Tprof2(T0,T1,Th1,Th2,Th3,w1,...
    w2,wh1,wh2,wh3,UA1,UA2,UA3,iter)

size=1/iter;

z=(size:size:1)';

R1=(w1-wh1)/(w1*wh1);
R2=(w2-wh2)/(w2*wh2);
R3=(w1-wh3)/(w1*wh3);

dT1 = (Th1-T0)*exp(R1*UA1.*z);
dT2 = (Th2-T0)*exp(R2*UA2.*z);
dT3 = (Th3-T1)*exp(R3*UA3.*z);
```

G.2 Perstorp Study

perstorp_ss.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%                                                                 %%%%%
%%%%%       PERSTORP STEADY-STATE CASE STUDY                          %%%%%
%%%%%                                                                 %%%%%
%%%%%           Daniel Greiner Edvardsen                              %%%%%
%%%%%                  IKP, NTNU                                      %%%%%
%%%%%                June 14, 2011                                    %%%%%
%%%%%                                                                 %%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Info: This m-file calculates the hot and cold outlet
% temperatures for a heat exchanger network with three
% streams; one the upper stream there are two heat
% exchangers in series, on the middle stream there is
% one heat exchanger and on the lower stream there is
% also one heat exchanger. The conditions are taken from
% Perstorp in Sweden.
% The optimum is found (optimal split) that
% results in the maximum end temperature. The SOC split
% is also found and compared with the optimal split and
% the actual split at Perstorp.

clc;
clear all;
close all;

% Which year? Alternatives: 2008.mat, 2009.mat, 2010.mat.
   Remember to change 'year'.mat in perstorpdata.m as well.
load 2010.mat;

% Which data point? k=1: first hour, k=8760: last hour
k=4000;

```

```
33 %% DEFINING PARAMETERS
34 % Loading relevant temperatures, mass streams, heat
       capacities, and densities from 'perstorpdata.m' All
       units in SI.
35 [T0 Tp Th11in Th1in Th2in Th3in Th11 Th1 Th2 Th3 T11_ T11
       T1 T2 T3 m0 mh11 mh1 mh2 mh3 m11 m1 m2 m3 cp0 cp11 cp1
       cp2 cp3 rho0 rho11 rho1 rho2 rho3 w0 w11 w1 w2 w3 wh11
       wh1 wh2 wh3] = perstorp_data(k);
36
37 %% UA AND SPLIT RATIOS
38
39 % Calculating UA-values from Perstorp Data
40 UA11 = w11*(T11-T0)/((Th11-T0)-(Th11in-T11))*log((Th11-T0)
       /(Th11in-T11));
41 UA1  = w1*(T1-T11)/((Th1-T11)-(Th1in-T1))*log((Th1-T11)/(
       Th1in-T1));
42 UA2  = w2*(T2-T0)/((Th2-T0)-(Th2in-T2))*log((Th2-T0)/(Th2in
       -T2));
43 UA3  = w3*(T3-T0)/((Th3-T0)-(Th3in-T3))*log((Th3-T0)/(Th3in
       -T3));
44
45 % Split ratios
46 ptm1 = m1/m0; ptm2 = m2/m0; ptm3 = m3/m0;
47 sum = ptm1+ptm2+ptm3;
48 pt1 = ptm1/sum; pt2 = ptm2/sum; pt3 = ptm3/sum;
49 P = [pt1 pt2 pt3];
50
51 %% ITERATION PROCEDURE
52
53 % Number of iterations
54 iter=100;
55
56 U1=1/iter:1/iter:1;
57 U2=1/iter:1/iter:1;
58
59 n = zeros(iter,iter);
```

```
60 T11=n; T1=n; T2=n; T3=n; Th11=n; Th1=n; Th2=n; Th3=n; Tmix=
       n; e11=n; eh11=n; e1=n; eh1=n; e2=n; eh2=n; e3=n; eh3=n;
        C11=n; C1=n; C2=n; C3=n; NTU11=n; NTU1=n; NTU2=n; NTU3=
       n; U=n; CON=n; CHEN=n; CHEN2=n; CON2=n;
61
62 for i=1:iter
63 for j=1:iter
64
65 u1 = i/iter;
66 u2 = j/iter;
67
68 if(u1+u2<1) % Assuring u3=1-u1-u2 is never = 0 (making NTU3
       = inf) nor < 0
69
70 % Calculating outlet temperatures and info about HEs (Only
       u1 and u2 are changing)
71 [T HE] = tempCalc3(T0,w0,UA11,UA1,UA2,UA3,Th11in,Th1in,
       Th2in,Th3in,wh11,wh1,wh2,wh3,u1,u2);
72
73 T11(i,j)=T(1); T1(i,j)=T(2); T2(i,j)=T(3); T3(i,j)=T(4);
       Th11(i,j)=T(5); Th1(i,j)=T(6); Th2(i,j)=T(7); Th3(i,j)=T
       (8); Tmix(i,j)=T(9);e11(i,j)=HE(1); eh11(i,j)=HE(2); e1(
       i,j)=HE(3); eh1(i,j)=HE(4); e2(i,j)=HE(5); eh2(i,j)=HE
       (6); e3(i,j)=HE(7); eh3(i,j)=HE(8);C11(i,j)=HE(9); C1(i,
       j)=HE(10); C2(i,j)=HE(11); C3(i,j)=HE(12); NTU11(i,j)=HE
       (13); NTU1(i,j)=HE(14); NTU2(i,j)=HE(15); NTU3(i,j)=HE
       (16);
74
75 else
76 break;
77 end
78
79 end
80 end
81
82 %% Results
```

```
83
84 [v p]=max(Tmix);
85 [v2 p2]= max(max(Tmix));
86
87 p1=p(p2);
88
89 % OPTIMAL (maximum) temperature
90 Tmax = Tmix(p1,p2);
91
92 % OPTIMAL splits
93 u1 = p1/iter;
94 u2 = p2/iter;
95
96 % Controlled variables
97 c1 = ((Th1in-T1)./(Th11in-T0)-1).*(T11-T0).^2./(Th1in-T11)
      +(T1-T0).^2./(Th1in-T11)-(T2-T0).^2./(Th2in-T0);
98 c2 = (T3-T0).^2./(Th3in-T0)-(T2-T0).^2./(Th2in-T0);
99
100 %% Plot Self-Optimizing solution
101 close all;
102 %How many contour lines? Default: 20
103 v=fix(Tmax)-20:1:fix(Tmax);
104 %How many labels? Defalt: Every 5th
105 v2=fix(Tmax)-20:5:fix(Tmax);
106
107 figure
108 contour(U1,U2,double(c1),[0 0],'b','linewidth',2); hold on;
109 contour(U1,U2,double(c2),[0 0],'r','linewidth',2); hold on;
110
111 [C,h] = contour(U1,U2,double(Tmix), v, 'HandleVisibility','
      off'); hold on;
112 th = clabel(C,h,v2);
113 set(th,'BackgroundColor',[1 1 .6],'Edgecolor',[.7 .7 .7])
114
115 [C2,h2]=contour(U1,U2,double(Tmix), round(Tp*10)./10, '
      linewidth',3,'HandleVisibility','off'); hold on;
```

```
th2 = clabel(C2,h2);
set(th2,'BackgroundColor',[0 1 0],'Edgecolor',[.7 .7 .7])

scatter(u2,u1,'k','filled');

xlabel('u_2'); ylabel('u_1');
legend('c_{1}=0' ,'c_{2}=0','T_{opt}');
grid on
```

perstorp_data.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%                                                                 %%%%%
%%%%%                      PERSTORP DATA                              %%%%%
%%%%%                                                                 %%%%%
%%%%%              Daniel Greiner Edvardsen                           %%%%%
%%%%%                       IKP, NTNU                                 %%%%%
%%%%%                    June 14, 2011                                %%%%%
%%%%%                                                                 %%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [T0 Tp Th11in Th1in Th2in Th3in Th11 Th1 Th2 Th3
    T11_ T11 T1 T2 T3 m0 mh11 mh1 mh2 mh3 m11 m1 m2 m3 cp0
    cp11 cp1 cp2 cp3 rho0 rho11 rho1 rho2 rho3 w0 w11 w1 w2
    w3 wh11 wh1 wh2 wh3 ] = perstorp_data(k)

load 2010.mat;

%%%%% Convert hours to seconds %%%%%
h=3600;

%%%%% All temperatures in 'Celsius' %%%%%%

T0 = ti1053y(k); %Feed temperature
Tp = ti2106y(k); %End temperature

```

```
23      % HOT temperature IN
24      Th11in  = ti2362y(k);
25      Th1in   = ti2303y(k);
26      Th2in   = fq2261y(k)/fq2064y(k) * ti2263y(k) +...
27                (1-fq2261y(k)/fq2064y(k))*ti2252y(k);
28      Th3in   = ti2352y(k);
29
30      % HOT Temperature OUT
31      Th11    = ti2021y(k);
32      Th1     = tc2043y(k);
33      Th2     = ti2031y(k);
34      Th3     = ti2011y(k);
35
36      % COLD Temperatyre OUT
37      T11_    = ti2024y(k);
38      T11     = ti2042y(k);
39      T1      = ti2044y(k);
40      T2      = ti2034y(k);
41      T3      = ti2014y(k);
42
43  %%%%% All heat capacities in 'J/kgK' %%%%%%
44
45  cp0       = (3728-3701.6)/50 ...
46            *(T0-50)+3701.6;
47  cp11    = 4006;
48  cp1     = 2827;
49  cp2     = 4216;
50  cp3     = (4166-4105)/60* ...
51            (T0-60)+4166;
52
53  %%%%%% Densities in 'kg/m^3' %%%%%%
54  rho0    = (1021-1054)/50*(T0-50)+1054;
55  rho1    = 1257;
56  rho2    = 926;
57  rho3    = (926-971.8)/60*(T0-60)+926;
58  rho11   = rho3;
```

```
59
60 %%%%%% Volumetric flow rates in 'm^3/s' %%%%%%
61
62 v0       = fc2105y(k)/h;
63 vh11     = fq2502y(k)/h;
64 vh1      = fc2304y(k)/h;
65 vh2      = fq2064y(k)/h;
66 vh3      = (fq2032y(k)-fq2558y(k))/h;
67
68     %Using energy balances to calculate cold volume streams
69     v11 = (Th11in-Th11)/(T11_-T0)*vh11*rho11/rho0*cp11/cp0;
70     v1  = (Th1in-Th1)/(T1-T11)*vh1*rho1/rho0*cp1/cp0;
71     v2  = (Th2in-Th2)/(T2-T0)*vh2*rho2/rho0*cp2/cp0;
72     v3  = (Th3in-Th3)/(T3-T0)*vh3*rho3/rho0*cp3/cp0;
73
74 %%%%%% Mass flow rates in 'kg/s' %%%%%%
75
76 m0       = v0  * rho0;
77 m11      = v11 * rho0;
78 m1       = v1  * rho0;
79 m2       = v2  * rho0;
80 m3       = v3  * rho0;
81
82 mh11     = vh11 * rho11;
83 mh1      = vh1  * rho1;
84 mh2      = vh2  * rho2;
85 mh3      = vh3  * rho3;
86
87 %%%%%% All mCps in 'W/K' (kg/s * J/kgK) %%%%%%
88
89 w0       = m0  * cp0;
90 w11      = m11 * cp0;
91 w1       = m1  * cp0;
92 w2       = m2  * cp0;
93 w3       = m3  * cp0;
94
```

```
95 wh11     = mh11 * cp11;
96 wh1      = mh1  * cp1;
97 wh2      = mh2  * cp2;
98 wh3      = mh3  * cp3;
```

tempCalc3.m

```
1  function [T HE] = tempCalc3(T0,w0,UA11,UA1,UA2,UA3,Th11in,
       Th1in,Th2in,Th3in,wh11,wh1,wh2,wh3,u1,u2)
2
3  u3=1-u1-u2;
4
5  w1 = u1*w0;
6  w2 = u2*w0;
7  w3 = u3*w0;
8
9  NTU11 = UA11/w1;
10 NTU1  = UA1/w1;
11 NTU2  = UA2/w2;
12 NTU3  = UA3/w3;
13
14 C11 = w1/wh11;
15 C1  = w1/wh1;
16 C2  = w2/wh2;
17 C3  = w3/wh3;
18
19 if(C11>0.99999 && C11<1.00001)
20     C11=0.99999;
21 elseif(C1>0.99999 && C1<1.00001)
22     C1=0.99999;
23 elseif(C2>0.99999 && C2<1.00001)
24     C2=0.99999;
25 elseif(C3>0.99999 && C3<1.00001)
26     C3=0.99999;
27 end
28
```

```
29 e11= (1-exp(-NTU11*(C11-1)))/(C11-exp(-NTU11*(C11-1)));
30 e1 = (1-exp(-NTU1*(C1-1)))/(C1-exp(-NTU1*(C1-1)));
31 e2 = (1-exp(-NTU2*(C2-1)))/(C2-exp(-NTU2*(C2-1)));
32 e3 = (1-exp(-NTU3*(C3-1)))/(C3-exp(-NTU3*(C3-1)));
33
34 eh11= e11*C11;
35 eh1 = e1*C1;
36 eh2 = e2*C2;
37 eh3 = e3*C3;
38
39 T11 = e11*Th11in + (1-e11)*T0;
40 T1  = e1*Th1in + (1-e1)*T11;
41 T2  = e2*Th2in + (1-e2)*T0;
42 T3  = e3*Th3in + (1-e3)*T0;
43
44 Th11= (1-eh11)*Th11in + eh11*T0;
45 Th1 = (1-eh1)*Th1in + eh1*T11;
46 Th2 = (1-eh2)*Th2in + eh2*T0;
47 Th3 = (1-eh3)*Th3in + eh3*T0;
48
49 Tmix = u1*T1+u2*T2+u3*T3;
50
51 T = [T11 T1 T2 T3 Th11 Th1 Th2 Th3 Tmix];
52 HE = [e11 eh11 e1 eh1 e2 eh2 e3 eh3 C11 C1 C2 C3 NTU11 NTU1
       NTU2 NTU3];
```

perstorp_dynamic.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%                                                                 %%%%%
%%%%%                  PERSTORP DYNAMIC STUDY                         %%%%%
%%%%%                                                                 %%%%%
%%%%%              Daniel Greiner Edvardsen                           %%%%%
%%%%%                       IKP, NTNU                                 %%%%%
%%%%%                    June 14, 2011                                %%%%%
%%%%%                                                                 %%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Info: This m-file simulates the dynamic simulink model
% perstorp_sim.mdl. The conditions are implemented in
% this file, the simulink file is then run and finally
% some plots are made in the end.

clc
clear all
load 2010.mat;

k=4000;

% Loading relevant temperatures, mass streams, heat
    capacities,
% and densities from 'perstorpdata.m'
[T0 Tp Th11in Th1in Th2in Th3in Th11 Th1 Th2 Th3 T11_ T11
    T1 T2 T3 m0 mh11 mh1 mh2 mh3 m11 m1 m2 m3 cp0 cp11 cp1
    cp2 cp3 rho0 rho11 rho1 rho2 rho3 w0 w11 w1 w2 w3 wh11
    wh1 wh2 wh3 ] = perstorp_data(k);

Th11in  = double(Th11in);
Th1in   = double(Th1in);
Th2in   = double(Th2in);
Th3in   = double(Th3in);

% ========================================================
```

```
32 %                MAIN HEAT EXCHANGERS
33 % ========================================================
34
35 % Heat transfer areas given from Perstorp [m^2]
36 A11 = 80;
37 A1  = 12.3;
38 A2  = 198;
39 A3  = 198;
40
41 % Volumes given from Perstorp [m^3]
42 Vh_tot = 0.775;      % Hot side volume [m3] for HE2 and HE3
43 Vc_tot = 0.760;      % Cold side volume [m3] for HE2 and HE3
44 Vh11_tot = 0.234;    % Hot side volume [m3] for HE11
45 Vc11_tot = 0.232;    % Cold side volume [m3] for HE11
46 Vh1_tot = 0.02226;   % Hot side volume [m3] for HE1
47 Vc1_tot = 0.02164;   % Cold side volume [m3] for HE1
48
49 N = 10;                  % Number of cells (Fixed)
50
51 % Estimation of heat transfer coefficients [W/m2*K]
52 h_h11 = 60.15;
53 h_h1  = 633;
54 h_h2  = 5615;
55 h_h3  = 5800;
56
57 h_c11 = h_h11;
58 h_c1  = h_h1;
59 h_c2 = h_h2;
60 h_c3 = h_h3;
61
62 % Hot side
63 Vh11 = Vh11_tot/N;  % Cell volume [m3] for HE11
64 Ah11 = A11/N;       % Cell area [m2] for HE11
65 Vh1 = Vh1_tot/N;    % Cell volume [m3] for HE1
66 Ah1 = A1/N;         % Cell area [m2] for HE1
67 Vh2 = Vh_tot/N;     % Cell volume [m3] for HE2
```

```
68 Ah2 = A2/N;           % Cell area [m2] for HE2
69 Vh3 = Vh_tot/N;      % Cell volume [m3] for HE3
70 Ah3 = A3/N;           % Cell area [m2] for HE3
71
72 % Cold side
73
74 Vc11 = Vc11_tot/N;
75 Ac11 = A11/N;
76 Vc1 = Vc1_tot/N;
77 Ac1 = A1/N;
78 Vc2 = Vc_tot/N;
79 Ac2 = A2/N;
80 Vc3 = Vc_tot/N;
81 Ac3 = A3/N;
82
83 % Wall
84 mass_w = 568;            % Weight of heat exchanger [kg]
85 rho_w  = 1240;           % Wall density [kg/m3] (TM10-BFG)
86 V_w    = mass_w/rho_w;   % Wall volume [m3]
87 cp_w   = 3000;           % Wall heat capacity [J/kg*K]
88 V_i_w  = V_w/N;          % Volume of one wall segment [m3]
89 A_i_w  = 200/N;          % Area of one wall segment [m2]
90
91
92
93 % ========================================================
94 %                   INITIAL CONDITIONS [ C]
95 % ========================================================
96
97 % Initial (temperature) conditions for the heat exchangers
98
99 % First column: Initial conditions, hot side (10)
100 % Second column: Initial conditions, wall (10)
101 % Third column: Initial conditions, cold side (10)
102
103 HX11init = [124.8292147 104.126289  57.63458142;
```

```
104              122.027725  101.2997531 60.51387896;
105              119.2228461 98.4697978  63.38969738;
106              116.4145738 95.63641881 66.26204087;
107              113.6029042 92.79961204 69.13091364;
108              110.787833  89.95937334 71.99631989;
109              107.9693562 87.11569856 74.85826379;
110              105.1474697 84.26858353 77.71674953;
111              102.3221693 81.41802411 80.5717813;
112              99.49345082 78.56401612 83.42336326];
113
114 HX1init   = [131.4011561 122.1805597 87.47503691
115              130.7910986 120.8480675 91.23232651
116              130.1332431 119.4111744 94.71662125
117              129.4238444 117.8617005 97.94775622
118              128.6588644 116.1908252 100.9441253
119              127.833948  114.3890367 103.7227861
120              126.9443994 112.4460778 106.2995566
121              125.9851545 110.3508879 108.6891057
122              124.9507527 108.0915396 110.9050364
123              123.8353054 105.6551712 112.9599633];
124
125 HX2init   = [138.8461623 134.3475021 59.03362354
126              126.0206234 122.0035326 63.82875206
127              114.5680237 110.9809515 69.19872121
128              104.3413937 101.1383079 75.21244297
129              95.2094967  92.34929258 81.9470905
130              87.05514455 84.50111752 89.48908846
131              79.77369369 77.49306833 97.93522213
132              73.2717024  71.2352118  107.3938794
133              67.46573165 65.64724185 117.9864418
134              62.28127433 60.65744894 129.848842];
135
136 HX3init   = [128.9568742 126.6540797 59.72703712
137              119.2940288 117.1471585 65.06361795
138              110.2854627 108.2839586 70.78778692
139              101.8868739 100.0208933 76.92769401
```

```
            94.05696035 92.31732685 83.51353367
            86.75721666 85.13537517 90.57769335
            79.95174458 78.43971929 98.15491273
            73.60707656 72.19743174 106.2824546
            67.69201118 66.37781456 115.0002881
            62.17745971 60.95224842 124.3512852];

% Controllers

tc10 = -0.634632447104129;
tc20 = 0.278170600005430;

%% Tuning parameters

% Controller 1
kc1 = 0.026;
tc1 = 62;
I1  = kc1/tc1;

% Controller 2
kc2 = 0.102;
tc2 = 174;
I2  = kc2/tc2;

%% Simulation

%Simulation time
simtime=3600;

sim('perstorp_sim',simtime);
systemHandle = get_param(gcs, 'Handle');
saveas(systemHandle , 'simulink.pdf', 'pdf')
%% PLOTS
close all;

%Convert from seconds to minutes
```

```
176 time=t/60;
177
178 figure
179 plot(time,T11(:,1),time,T1,time,T2,time,T3,time,Tend,'
       LineWidth',2)
180 % Title('Cold side outlet temperatures');
181 legend('T11','T1','T2','T3','Tend')
182 xlabel('Time [min]');
183 ylabel('Temperature [\circC]')
184 axis([0 15 80 140]);
185
186 figure
187 plot(time,Th11,time,Th1,time,Th2,time,Th3)
188 % Title('Hot side outlet temperatures');
189 legend('Th11','Th1','Th2','Th3')
190 xlabel('Time [min]');
191 ylabel('Temperature [\circC]')
192
193 figure
194 plot(time,u1,time,u2,time,u3,'LineWidth',2);
195 % Title('Mass fractions');
196 axis([0 10 0 0.7]);
197 xlabel('Time [min]'); ylabel('Splits');
198 legend('u_1', 'u_2', 'u_3');
199 grid on;
200
201 figure
202 plot(time,c1,time,c2,'LineWidth',2);
203 xlabel('Time [min]'); ylabel('Controlled variables');
204 % axis([0 time(end) -1 1]);
205 legend('c_1', 'c_2');
206 grid on;
```

G.3 Crude Unit Heat Exchanger Network

mongstadSolve.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%                                                                 %%%%%
%%%%%                      MONGSTAD STUDY                             %%%%%
%%%%%                                                                 %%%%%
%%%%%              Daniel Greiner Edvardsen                           %%%%%
%%%%%                        IKP, NTNU                                %%%%%
%%%%%                      June 14, 2011                              %%%%%
%%%%%                                                                 %%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Info: This m-file calculates the hot and cold outlet
% temperatures for a heat exchanger network with seven
% streams in parallel. The conditions are taken from
% Statoil Mongstad. The SOC split is found and compared
% to the real RTO split.

clear all;
clc;

%% Defining all parameters

global T0;
global m0
global Tcold;
global Thot;

T0  = 132.909;
%% Initial flows (RTO)

m0A = 24.08753; % Stream A
m0B = 48.49833; % Stream B
m0C = 17.84011; % Stream C
m0D = 35.61500; % Stream D
```

```
34 m0E = 34.68861; % Stream E
35 m0F = 65.57833; % Stream F
36 m0G = 28.73583; % Stream G
37
38 % Total flow minus m0G
39 m0=m0A+m0B+m0C+m0D+m0E+m0F;
40
41 % Initial splits (without stream G)
42 u0=[m0A m0B m0C m0D m0E m0F]./m0;
43
44 %% RTO results (0 ('zero') indicates RTO)
45 [c,ceq0]=constraints(u0);
46 Tcold0=Tcold;
47 Thot0=Thot;
48
49 T1A0 = Tcold0(1); T2B0 = Tcold0(3); T2C0 = Tcold0(5);
50 T2D0 = Tcold0(7); T1E0 = Tcold0(8); T2F0 = Tcold0(10);
51
52 Tend0 = u0(1)*T1A0+u0(2)*T2B0+u0(3)*T2C0+u0(4)*T2D0+u0(5)*
      T1E0+u0(6)*T2F0;
53
54 %% Running fmincon to find optimal solution
55 options = optimset('Display','iter');
56
57 % Function to be minimized, J = 1
58 % Results: fmincon finds the 'u' that fulfils the
      constraints
59
60 u = fmincon(@(x)1,u0,[],[],[],[],0,1,@constraints,options);
61
62 % Calculating (intermediate) end temperature
63
64 Th1A = Thot(1); Th1B = Thot(2); Th2B = Thot(3); Th1C = Thot
      (4); Th2C = Thot(5); Th1D = Thot(6); Th2D = Thot(7);
      Th1E = Thot(8); Th1F = Thot(9); Th2F = Thot(10);
```

```
65 T1A = Tcold(1); T1B = Tcold(2); T2B = Tcold(3); T1C = Tcold
       (4); T2C = Tcold(5); T1D = Tcold(6); T2D = Tcold(7); T1E
        = Tcold(8); T1F = Tcold(9); T2F = Tcold(10);
66
67 Tend = u(1)*T1A+u(2)*T2B+u(3)*T2C+u(4)*T2D+u(5)*T1E+u(6)*
       T2F;
68
69 % New mass streams
70 m=u.*m0;
71 m(7) = m0G;
72
73 % Splits including stream G
74 unew = m./sum(m); % SOC splits
75 u0new=[m0A m0B m0C m0D m0E m0F m0G]./sum(m); % RTO splits
76
77 %% Temperature after 01
78
79 % Temperature after heat exchanger 01 (E153)
80 wc      = 615377.76;
81 n       = 1;
82 wh      = 186285.21;
83 Thin    = 282.608;
84 UA      = 144051.68;
85
86 [T Th] = tempCalc(Tend,Thin,wc,wh,UA,n);
87
88 % Temperature after mixing with stream G
89 cp0     = 2406.224915;
90 wc1G    = m0G*cp0;
91 n1G     = 1;
92 wh1G    = 289894.73;
93 Th1inG  = 167.487;
94 UA1G    = 96449.75;
95
96 [T1G Th1G] = tempCalc(T0,Th1inG,wc1G,wh1G,UA1G,n1G);
97
```

```
98  wc2G     = m0G*2622.608824;
99  n2G      = 1;
100 wh2G     = 317501.17;
101 Th2inG   = 262.586;
102 UA2G     = 125079.36;
103
104 [T2G Th2G] = tempCalc(T1G,Th2inG,wc2G,wh2G,UA2G,n2G);
105
106 TendFinal = (1-unew(7))*T + unew(7)*T2G;
107
108 Tfinal  = [T1A T2B T2C T2D T1E T2F T2G Tend TendFinal]';
109
110 %% RTO temperature
111
112 % Temperature after heat exchanger 01 (E153)
113
114 [Trto Thrto] = tempCalc(Tend0,Thin,wc,wh,UA,n);
115
116 % Temperature after mixing with stream G
117 TendFinal0 = (1-u0new(7))*Trto + u0new(7)*T2G;
118
119 Tfinal0 = [T1A0 T2B0 T2C0 T2D0 T1E0 T2F0 T2G Tend0
      TendFinal0]';
```

constraints.m

```
1  function [c,ceq] = constraints(u)
2
3  % Mass flows (m) in kg/s
4  % Temperatures in Celcius
5  % mCp's in W/K
6  % UA's in W/K
7  % n = number of shells
8
9  global T0;
10 global m0;
```

```
11 global T1; global Th1; global T2; global Th2;
12
13 %% Parameters for stream A
14
15 m0A      = m0*u(1);
16 cp0A     = 2570.944432;
17 wcA      = m0A*cp0A;
18
19 nA       = 4;
20 whA      = 48187.09;
21 Th1inA   = 300.416;
22 UAA      = 131025.48;
23
24 [T1A Th1A] = tempCalc(T0,Th1inA,wcA,whA,UAA,nA);
25
26 %% Parameters for stream B
27
28 m0B      = m0*u(2);
29
30 cp0B1    = 2445.743628;
31 wc1B     = m0B*cp0B1;
32
33 wh1B     = 92541.46;
34 nB1      = 2;
35 UA1B     = 102709.10;
36
37 cp0B2     = 2618.432483;
38 wc2B     = m0B*cp0B2;
39 nB2      = 2;
40 wh2B     = 129499.93;
41 UA2B     = 88644.82;
42
43 Th2inB   = 270.254;
44
45 tparB = [T0 wc1B wh1B UA1B nB1 Th2inB wc2B wh2B UA2B nB2
      1];
```

```
46
47 %Using fsolve to find Th1in=Th2
48 Th1inB  = fsolve(@(x) FindTh1in(x,tparB),232,optimset('
      Display','off'));
49
50 % Temperatures calculated in the function FindTh1in
51 T1B = T1; Th1B = Th1; T2B = T2; Th2B = Th2;
52
53 %% Parameters for stream C
54
55 m0C      = m0*u(3);
56
57 cp0C1    = 2407.952102;
58 wc1C     = m0C*cp0C1;
59
60 wh1C     = 35816.4165;
61 UA1C     = 84642.10;
62 nC1      = 2;
63
64 cp0C2    = 2584.355153;
65 wc2C     = m0C*cp0C2;
66 wh2C     = 38747.8648;
67 UA2C     = 133605.46;
68 nC2      = 3;
69
70 Th2inC   = 245.375;
71
72 tparC = [T0 wc1C wh1C UA1C nC1 Th2inC wc2C wh2C UA2C nC2
      2];
73
74 %Using fsolve to find Th1in=Th2
75 Th1inC  = fsolve(@(x) FindTh1in(x,tparC),175.5,optimset('
      Display','off'));
76
77 % Temperatures calculated in the function FindTh1in
78 T1C = T1; Th1C = Th1; T2C = T2; Th2C = Th2;
```

```
79
80 %% Parameters for stream D
81
82 m0D      = m0*u(4);
83 cp0D1    = 2483.068751;
84 wcD      = m0D*cp0D1;
85
86 n1D      = 4;
87 wh1D     = 118477.94;
88 Th1inD   = 226.181;
89 UA1D     = 132831.92;
90
91 [T1D Th1D] = tempCalc(T0,Th1inD,wcD,wh1D,UA1D,n1D);
92
93 cp0D2    = 2649.127849;
94 wc2D     = m0D*cp0D2;
95 n2D      = 1;
96 wh2D     = 33935.43;
97 Th2inD   = 273.813;
98 UA2D = 41563.90;
99
100 [T2D Th2D] = tempCalc(T1D,Th2inD,wc2D,wh2D,UA2D,n2D);
101
102 %% Parameters for stream E
103
104 m0E      = m0*u(5);
105 cp0E     = 2527.081131;
106 wcE      = m0E*cp0E;
107
108 nE       = 4;
109 whE      = 79890.13;
110 Th1inE   = 256.408;
111 UAE      = 190981.94;
112
113 [T1E Th1E] = tempCalc(T0,Th1inE,wcE,whE,UAE,nE);
114
```

```
115 %% Parameters for stream F
116
117 m0F      = m0*u(6);
118 cp0F1    = 2382.242363;
119 wcF      = m0F*cp0F1;
120
121 n1F      = 1;
122 wh1F     = 47248.68;
123 Th1inF   = 203.201;
124 UA1F     = 49437.00;
125
126 [T1F Th1F] = tempCalc(T0,Th1inF,wcF,wh1F,UA1F,n1F);
127
128 cp0F2    = 2535.118619;
129 wc2F     = m0F*cp0F2;
130 n2F      = 4;
131 wh2F     = 175073.77;
132 Th2inF   = 248.07;
133 UA2F     = 224053.26;
134
135 [T2F Th2F] = tempCalc(T1F,Th2inF,wc2F,wh2F,UA2F,n2F);
136
137
138 %% Saving temperatures
139 global Tcold
140 global Thot
141
142 Tcold=[T1A T1B T2B T1C T2C T1D T2D T1E T1F T2F];
143 Thot=[Th1A Th1B Th2B Th1C Th2C Th1D Th2D Th1E Th1F Th2F];
144
145 %% Controlled variables
146 fA = (T1A-T0).^2./(Th1inA-T0);
147 fB = ((Th2inB-T2B)./(Th1inB-T0)-1).*(T1B-T0).^2./(Th2inB-
      T1B)+(T2B-T0).^2./(Th2inB-T1B);
148 fC = ((Th2inC-T2C)./(Th1inC-T0)-1).*(T1C-T0).^2./(Th2inC-
      T1C)+(T2C-T0).^2./(Th2inC-T1C);
```

```
149 fD = ((Th2inD-T2D)./(Th1inD-T0)-1).*(T1D-T0).^2./(Th2inD-
      T1D)+(T2D-T0).^2./(Th2inD-T1D);
150 fE = (T1E-T0).^2./(Th1inE-T0);
151 fF = ((Th2inF-T2F)./(Th1inF-T0)-1).*(T1F-T0).^2./(Th2inF-
      T1F)+(T2F-T0).^2./(Th2inF-T1F);
152
153 cA = fA - fF;
154 cB = fB - fF;
155 cC = fC - fF;
156 cD = fD - fF;
157 cE = fE - fF;
158
159 %% Constraints
160 c=[]; %No inequality constraints
161
162 % Equality constraints = 0
163 ceq=[cA
164      cB
165      cC
166      cD
167      cE
168      sum(u)-1];
```

tempCalc.m

```
1  function [T1 Th1] = tempCalc(T0,Th1in,wc,wh,UA,n)
2
3  Cmin = min(wc,wh);
4  Cmax = max(wc,wh);
5
6  Cr = Cmin/Cmax;
7
8  NTU  = UA/Cmin;
9  NTU1 = NTU/n;
10
11 scr=sqrt(1+Cr^2);
```

```
12 a=1+exp(-NTU1*scr);
13 b=1-exp(-NTU1*scr);
14
15 e1 = 2/(1+Cr+scr*a/b);
16
17 %For only one shell pass
18 if(n==1)
19     e=e1;
20 end
21
22 %For n>1 Shell passes
23 if(n>1)
24     x = ((1-e1*Cr)/(1-e1))^n-1;
25     y = ((1-e1*Cr)/(1-e1))^n-Cr;
26
27     e = x/y;
28 end
29
30 % Cold temperature out
31 T1 = Cmin/wc*e*Th1in + (1-Cmin/wc*e)*T0;
32
33 % Hot temperature out
34 Th1 = (1-Cmin/wh*e)*Th1in + Cmin/wh*e*T0;
```

FindTh1in.m

```
1 function F = findTh1in(Tin,tpar)
2
3 Th1in=Tin;
4
5 T0      = tpar(1);
6 wc      = tpar(2);
7 wh1     = tpar(3);
8 UA1     = tpar(4);
9 n1      = tpar(5);
10 Th2in   = tpar(6);
```

```
wc2     = tpar(7);
wh2     = tpar(8);
UA2     = tpar(9);
n2      = tpar(10);
stream  = tpar(11);

global T1; global Th1;
[T1 Th1] = tempCalc(T0,Th1in,wc,wh1,UA1,n1);


global T2; global Th2;
[T2 Th2] = tempCalc(T1,Th2in,wc2,wh2,UA2,n2);

if(stream==1)
    F = Th1in-Th2;
elseif(stream==2)
    F = (Th1in+1.205)-Th2;
end
```

H Maple Code

H.1 Two Heat Exchangers in Parallel

```
restart;
with(LinearAlgebra):with(VectorCalculus):

# Model equations
g1:= u*w0*T1+(1-u)*w0*T2-w0*Tend:
g2:= 2*u*w0*(T1-T0)-UA1*(Th1in-T1+Th1-T0):
g3:= 2*(1-u)*w0*(T2-T0)-UA2*(Th2in-T2+Th2-T0):
g4:= 2*wh1*(Th1-Th1in)+UA1*(Th1in-T1+Th1-T0):
g5:= 2*wh2*(Th2-Th2in)+UA2*(Th2in-T2+Th2-T0):

g:=[g1,g2,g3,g4,g5]:

# z = MV's (split u) and state variables
z:=[u,T1,T2,Tend,Th1,Th2]:

gradG:= Jacobian(g,z):
N:=NullSpace(gradG):
N:=Matrix([N[1]]):
NT:=Transpose(N):

# Objective function
J:=[-Tend]:

gradJ:=Transpose(Jacobian(J,z));



# Calculating reduced gradient
Jzred:=MatrixMatrixMultiply(NT,gradJ):

```

```
31 # Simplifying
32 Jzred:=simplify(Jzred):
33 Jzred:=simplify(numer(Jzred[1,1])):
34 Jzred:=factor(Jzred);
35
36 # Loading 'multires' package
37 read("multires.mpl"):
38
39 # Variables to be eliminated
40 varlist:=[u,w0,wh1,wh2];
41
42 # Arguments for the sparse resultant
43 polylist:=[Jzred,g2,g3,g4,g5];
44
45 #Sparse resultant
46 R:= det(spresultant(polylist,varlist));
47 R:=factor(R);
48
49 # Simplifying
50 T0:=0;
51
52 # Self-optimizing variable = 0
53 c:=R;
54
55 save Jzred,R,c, "2HXpar";
```

H.2 Two Heat Exchangers in Series and One in Parallel

```
> restart;
> with(LinearAlgebra):with(VectorCalculus):
>
> DT11:= Th11in-T11+Th11-T0:
> DT1 := Th1in-T1+Th1-T11:
> DT2 := Th2in-T2+Th2-T0:
>
> g1:=   w0*Tend-u*w0*T1-(1-u)*w0*T2:
> g2:=   2*u*w0*(T11-T0)-UA11*DT11:
> g3:=   2*wh11*(Th11-Th11in)+UA11*DT11:
> g4:=   2*(1-u)*w0*(T2-T0)-UA2*DT2:
> g5:=   2*wh2*(Th2-Th2in)+UA2*DT2:
> g6:=   2*u*w0*(T1-T11)-UA1*DT1:
> g7:=   2*wh1*w0*(Th1-Th1in)+UA1*DT::
>
> g:=[g1,g2,g3,g4,g5,g6,g7];
>
> # z = MV's (split u) and state variables
> z:=[u,T11,T1,T2,Tend,Th11,Th1,Th2]:

> gradG:= Jacobian(g,z):
> N:=NullSpace(gradG):
> N:=Matrix([N[1]]):
> N:=Transpose(N):
>
> # Objective function
> J:=[-Tend]:
>
> gradJ:=Transpose(Jacobian(J,z)):
>
> # Calculating reduced gradient
> Jzred:=MatrixMatrixMultiply(N,gradJ):
```

```
> 
> # Simplifying
> Jzred:=Jzred[1,1]:
> Jzred:=simplify(Jzred):
> Jzred:=simplify(numer(Jzred)):
> Jzred:=factor(Jzred);
> 
> # Loading 'multires' package
> read("multires.mpl"):
> 
> # Variables to be eliminated
> varlist:=[u,w0,wh11,wh2,wh1,UA11,UA1]:
> 
> # Arguments for the sparse resultant
> polylist:=[Jzred,g1,g2,g3,g4,g5,g6,g7]:
> 
> #Sparse resultant
> R:=det(spresultant(polylist,varlist)):
> R:=factor(R);
> 
> # Simplifying
> T0:=0:
> 
> # Self-optimizing variable = 0
> c:=R;

> save Jzred,R,c, "2HXser1HXpar";
```

I Risk Assessment

NTNU	HSE action plan	Prepared by	Number	Date
HSE		The HSE section	HMSRV-12/24	01.12.2006
		Approved by	Page	Replaces
		The Rector	1 of 1	20.08.1999

Unit: Department of Chemical Engineering

What	Measure	Unit responsible	Priority	Cost	Current status
Programming and simulation does not need any HSE action plan					

Date: ____________________ Line manager: ____

NTNU HSE	Hazardous activity identification process	Prepared by	Number	Date
		HSE section	HMSRV-26/01	01.12.2006
		Approved by	Page	Replaces
		The Rector	1 out of 1	15.12.2003

Unit: Department Chemical Engineering **Date:** 16.06.2011
Participants in the identification process (including their function):

Short description of the main activity/main process: Simulation and programming on computer.

Activity/process	Responsible person	Laws, regulations etc.	Existing documentation	Existing safety measures	Comment
Simulation and programming on computer	S. Skogestad	-	-	-	Not any safety issues

NTNU HSE/KS	Risk assessment	Prepared by HSE section Approved by The Rector	Number HMSRV-26/03 Page 1 out of 2	Date 01.12.2006 Replaces 15.12.2003

Unit: Department of Chemical Engineering **Date:** 16.06.2011
Line manager: Øyvind Gregeresen
Participants in the risk assessment (including their function): Daniel Greiner Edvardsen

Activity from the identification process form	**Potential undesirable incident/strain**	**Likelihood:**	**Consequence:**			**Risk value**	**Comments/status Suggested measures**
		Likelihood (1-4)	Human (1-4)	Environment (1-4)	Economy/ materiel (1-4)		
Working with PC; simulation and programming.	Tear on back and arms.	1	1	1	1	1	This work has not been dangerous to the student or other. Work position can make tear on arms and back. Working on computers can give damage on eyes.

Likelihood, e.g.:
1. *Minimal*
2. *Low*
3. *High*
4. *Very high*

Consequence, e.g.:
1. *Relatively safe*
2. *Dangerous*
3. *Critical*
4. *Very critical*

Risk value (each one to be estimated separately):
Human = Likelihood x Human Consequence
Environmental = Likelihood x Environmental consequence
Financial/material = Likelihood x Consequence for Economy/materiel

Potential undesirable incident/strain
Identify possible incidents and conditions that may lead to situations that pose a hazard to people, the environment and any materiel/equipment involved.

Criteria for the assessment of likelihood and consequence in relation to fieldwork
Each activity is assessed according to a worst-case scenario. Likelihood and consequence are to be assessed separately for each potential undesirable incident. Before starting on the quantification, the participants should agree what they understand by the assessment criteria:

The likelihood of something going wrong is to be assessed according to the following criteria:

1 **Minimal**
Once every 10 years or less

2 **Low**
Once a year

3 **High**
Once a month

4 **Very high**
Once a week or more often

Human consequence is to be assessed according to the following criteria:

1 **Relatively safe**
Injury that does not involve absence from work; insignificant health risk

2 **Dangerous**
Injury that involves absence from work; may produce acute sickness

3 **Critical**
Permanent injury; may produce serious health damage/sickness

4 **Very critical**
Injury that may produce fatality/ies

Environmental consequences are assessed according to the following criteria:

1 **Relatively safe**
Insignificant impact on the environment

2 **Dangerous**
Possibility of undesirable long term effects; some cleanup is to be expected

3 **Critical**
Undesirable long term effects; cleanup to be expected

4 **Very critical**
Damaging to living organisms; irreversible impact on the environment; cleanup must be undertaken

The unit makes its own decision as to whether opting to fill in or not consequences for economy/materiel, for example if the unit is going to use particularly valuable equipment. It is up to the individual unit to choose the assessment criteria for this column.

Risk = Likelihood x Consequence
Please calculate the risk value for “Human”, “Environment” and, if chosen, “Economy/materiel”, separately. For activities with a risk value of 16 or 12, or a single value of 4, safety measures (designed to both reduce the likelihood and to limit the consequences) must be documented with descriptions of measures and allocation of responsibility.

About the column ”Comments/status, suggested preventative and corrective measures”:
Measures can impact on both likelihood and consequences. Prioritise measures that can prevent the incident from occurring; in other words, likelihood-reducing measures are to be prioritised above greater emergency preparedness, i.e. consequence-reducing measures.